BASKENT UNIVERSITY

INSTITUTE OF SCIENCE ENGINEERING

USE OF ARTIFICIAL INTELLIGENCE SYSTEMS IN CONTROL AND TRAFFIC FLOW GUIDANCE OF THE GROUP ELEVATORS

ALI BERKOL

MASTER OF SCIENCE, THESIS

2013

GRUP ASANSÖRLERİN DENETİMİNDE VE TRAFİK AKIŞININ YÖNLENDİRİLMESİNDE AKILLI SİSTEMLERİN KULLANILMASI

USE OF ARTIFICIAL INTELLIGENCE SYSTEMS IN CONTROL AND TRAFFIC FLOW GUIDANCE OF THE GROUP ELEVATORS

ALİ BERKOL

Thesis Submitted
in Partial Fulfillment of the Requirements
For the Degree of Master of Science
in Department of Electrical and Electronics Engineering
at Başkent University

2013

This thesis, titled: “Use of Artificial Intelligence Systems in Control and Traffic Flow Guidance of the Group Elevators”, has been approved in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONICS ENGINEERING, by our jury, on 21/06/2013.

Chairman :

Assoc. Prof Dr. Hasan OĞUL

Member (Supervisor) :

Assoc. Prof. Dr. Hamit ERDEM

Member :

Asst. Prof Dr. Derya YILMAZ

APPROVAL

…./06/2013

Prof. Dr. Emin AKATA
Director
Institute of Science and Engineering

This thesis is dedicated to;

My Mother; Füsun Berkol
My Father; Atilla Berkol
My Fiance; Lale Deniz Çokbilen
And
More than an advisor; Dr. Hamit Erdem

ÖZ

GRUP ASANSÖRLERİN DENETİMİNDE VE TRAFİK AKIŞININ YÖNLENDİRİLMESİNDE AKILLI YÖNTEMLERİN KULLANILMASI

Ali Berkol
Başkent Üniversitesi Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Günümüzde yüksek katlı binalarda çoklu asansörlerin devreye girmesi, kalabalık bina sakinlerinin en seri ve minimum enerji ile nasıl taşınabileceği problemini beraberinde getirmiştir. Grup Asansör Kontrol Sistemi, iki veya daha fazla sayıda asansör kabininin en uygun enerji, zaman ve talep dengesi gözetilerek sistematik olarak yönetilmesidir. Kabinin, katlardan gelen her bir çağrıya yanıt verirken bekleme zamanı, iniş-çıkış süreleri, enerji tüketimi, kullanıcı için kullanım kolaylığı vb. en ideal yönlendirme ile çalıştırılması amaçlanmıştır. Akıllı Sistemler, yukarıda bahsedilen parametrelerin en uygun değerlemelerini yapabilen esnek hesaplama yöntemi sunmaktadır. Mühendislik tasarım problemi olarak bakıldığında bekleme ve ulaşım süresinin ayarlanması, bu hizmet verilirken minimum enerji harcanması bir eniyileme problemidir. Bu problem karmaşık bir problem olduğu için çözüme yönelik akıllı sistem tabanlı yöntemlerin uygulanması uygun görülmüştür. Bu çalışmada genetik algoritma ve yapay sinir ağlarından oluşan hybrid bir yöntem uygulanarak sistemin performansını arttırmak amaçlanmıştır. Bu çalışma, 20 katlı binanın 4 kabinli asansör sisteminde simule edilmiş ve benzer çalışmalarla karşılaştırılmıştır.

ANAHTAR SÖZCÜKLER: Asansör, Asansör Grup Kontörlü, Yapay Sinir Ağları, Genetik Algoritma, Optimizasyon

Danışman: Doç.Dr. Hamit Erdem, Başkent Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü.

ABSTRACT

USE OF ARTIFICIAL INTELLIGENCE SYSTEMS IN CONTROL AND TRAFFIC FLOW GUIDANCE OF THE GROUP ELEVATORS

Ali Berkol
Instıtute of Science Engineering
Electrical and Electronics Department

Nowadays highrise buildings with multicar elevator systems bring the question of optimal car movements as how to convey the big populations of residences in the most quickest and efficient way. The Elevator Group Controller is a control system that accomplishes the systematic management of two or more elevators so that the elevator system is operated at best balance of energy, time and demand. The objective is that, elevator cars shall be assigned correspondingly in response to hall calls, so as to optimize waiting time, travel time, power consumption, passengers' comfort, etc. Artificial Intelligence presents soft computing options in designing a controller that capable of solving aforementioned numerous objectives. In the engineering design perspective, regulation of average waiting time and travel time with minimum energy consumption is an optimization problem. Since this is a complex problem, intelligent system based methods are chosen to be suitable for the solution of the same. This thesis aims to improve the system performance by applying a hybrid method that comprise of genetic algorithm and artificial neural network. This study simulates 20-floor building with a 4-multicar system and compares with the similiar studies.

KEY WORDS: Elevator, Elevator Group Control, Neural Network, Genetic Algoritm, Artificial Intelligence, Optimization.

Advisor: Associated Prof. Dr. Hamit Erdem, Başkent University, Electrical and Electronics Department.

TABLE OF CONTENTS

Page

LIST OF ABBREVIATIONS

MCES	Multicar Elevator Systems
EGCS	Elevator Group Control Systems
GNP	Genetic Network Programming
DFGS	Destination Floor Guidance System
DDES	Double Deck Elevator Systems
GA	Genetic Algorithm
GP	Genetic Programming
EC	Energy Consumption
AWT	Average Waiting Time
LWP	Long Waiting Percentage
ATT	Average Travel Time
IQS	Interconnected Queue Selective
IDC	Interconnected Down Collective
IFC	Interconnected Full Collective
RL	Reinforcement Learning
ANN	Artificial Neural Network

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1. INTRODUCTION

1.1 Project Overview

Today's urban life cannot be imagined without elevators. An elevator system is a system that transports passengers from one floor to another in a building. As vertical transportation vehicles, elevators were invented to ease the movement of people from one floor to another by shortening the time spent for the movement and by reducing the use of much human physical energy to climb up and down through the stairs. As many tall buildings with increasing number of floors were constructed in recent years, the need for better transportation services has become an important issue in the elevator industry. More elevators were installed, and a group control system was employed to coordinate the operation of individual elevators in a group manner. To enhance the effectiveness and efficiency of the Elevator Group Control Systems (EGCS) various control strategies and dispatching algorithms were developed over the years by using a variety of techniques. Passengers are transported in respond to their requests, which consists of hall calls and car calls. Passengers often have a long wait for the next elevator because they missed an elevator that left a few seconds previously; passengers often have to wait for the door to close even if no one is going to board. A passenger who wants to go to another floor from the current floor presses a direction button (hall call) and waits for an elevator to arrive, then enters the elevator and presses a floor button (car call) in the elevator. Basically, an elevator system is controlled by a two level control hierarchy that must solve two different control problems. The lower level task is to command each elevator to move up or down, to stop or start and to open and close the door. The higher level coordinates the movement of a group of elevators through a set of logical rules crafted to improve the system performance. This problem is solved by means of a group control system with the aid of a group supervisory control strategy. [3]

Control systems comption problem the optimization of energy, travel time and waiting time. Related elevator problems, such as average waiting time, travel time, energy

consumption, traffic flow, enter into the engineering concern where engineers worked in seeking proper solutions addressing to it.
Optimization of waiting time and energy consumption in particular is regarded as one of the basic problems of the elevator supervisory control and has been subject of miscellaneous studies so far. Multicar Elevator, MCE, systems introduce the idea of operating several elevator cars within a single shaft. This feature provides the advantage of saving space within the building while maintaining the transportation capacity level of conventional single-car elevators. ThyssenKrupp, a German elevator manufacturer, implemented the first MCE system, consisting of two elevator cars per shaft, at Stuttgart University in 2002. MCE group control method function to minimize the expected schedule completion time, and idle-car parking strategies to improve the call-allocation performance. With respect to the increase in the number of cars, algorithm achieved a greater reduction in the service completion time than the one of a method applying the direction coordination approach. [2]

To apply high level solutions, many simple heuristic approaches have been developed, such as collective control in which a car stops to serve the nearest call in its current movement direction, the longest queue first, and the highest unanswered floor first. Advanced intelligent technologies such as expert system, fuzz logic, artificial neural network, and reinforcement learning have been used to develop intelligent elevator scheduling methods. [7] Genetic network programming (GNP), one of the evolutionary computations, can realize a rule-based MCES due to its directed graph structure of the individual, which makes the system more flexible. In a paper, they use “The destination floor guidance system” (DFGS). [4] Some studies with Double-deck elevator system (DDES) is one of the solutions, where two cages are connected vertically in each shaft. [5] In an another study, the self-tuning FLC only exploits the waiting time for its tuning. [6] In another study, the multi-elevator system under consideration is an idealized system. It consists of k elevators traveling at unit speed. Stopping, entering, exiting, starting, and turning may take individual additional time, but we will not mention this in the sequel, for the ease of exposition. Each elevator has capacity one, i.e., it can only carry one request at a time. The difference in the literature between an online-algorithm and a (control) policy is that an online-algorithm has never any information about the input distribution, whereas a policy may know the input distribution. Moreover, online algorithms are usually evaluated by competitive analysis, which measures their worst-case deviation from an a-posteriori offline-optimum solution on finite input sequences, whereas policies are usually evaluated by expected performance w.r.t. to an input distribution, an average measure. Nearest-Neighbor minimizes the length of the server's empty move to the next request. For unit speed, this is the same as minimizing the start time of the next loaded move. [10] For example, in figure 1.1 shows a typical elevator group control mechanism for an 8-floor building.

Passenger arrivals are random events, where actual passenger arrival rates in buildings can be assumed to follow the Poisson distribution.
Four criteria are used as performance measures;

- Average waiting time (AWT) is the average time until the service elevator arrives at the floor after a passenger presses a hall call button.
- Average Traveling Time (ATT) is the average time until a passenger drops off at the destination floor after he/she gets into the car.
- Long Waiting Percentage (LWP) is the percentage of the passengers waiting over 60 s after a passenger presses a hall call button until he/she enters into a car.
- Energy Consumption (EC) is the total energy of all the elevators during their operation.

In the typical office buildings, the traffic is divided into three traffic patterns;

- Regular Traffic: during the work time, passengers move up and down among several different floors.
- Up-peak Traffic: typically in the morning when people arrive for work, most of the passengers travel from the lobby to the upper floors.
- Down-peak Traffic: at the end of the day, most of the passengers leave the floors and travel primarily to the lobby in order to exit the building.

Regarding this traffic follow a software has been expanded.

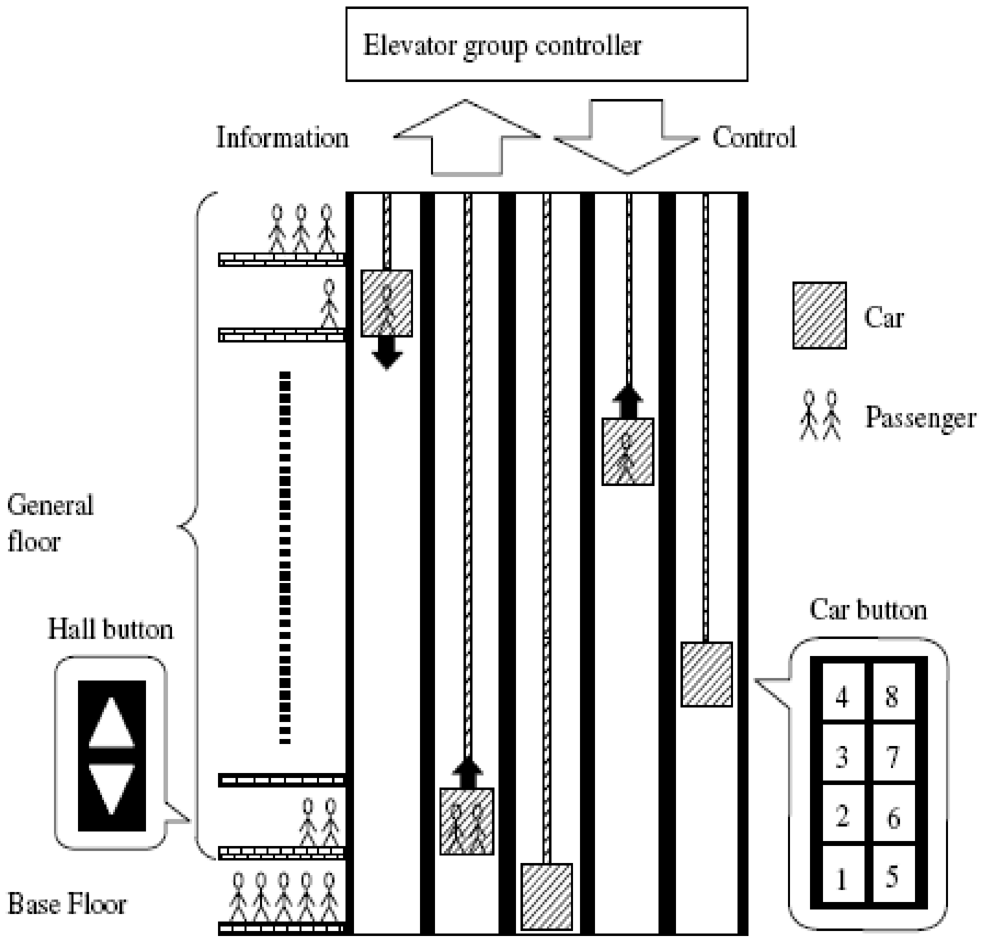

Figure 1.1: A typical EGC mechanism for an 8-floor building.

In another study with reinforcement learning, therefore an intelligent elevator control system has been applied to solve this problem of optimization of multiple objectives. However, an optimum solution to this problem is not known yet, or perhaps not possible due to its stochastic nature. In Reinforcement Learning (RL) algorithms have been demonstrated to be powerful heuristic methods for addressing large-scale control problems and have been applied to elevator group control. A team of RL agents, each of which is responsible for controlling one elevator car was used. The team receives a global reward signal which appears noisy to each agent due to the effects of the actions of the other agents, the random nature of the arrivals and the incomplete observation of the state. The results demonstrated the power of multi-agent RL on a very large scale stochastic dynamic optimization problem of practical utility. This study optimize the passenger waiting time.[18]

Along with the other studies, this thesis incorporates offline and online stages by using hybrid algorithms to improve to optimize average waiting time, total travel time, energy consumption. The algorithm shall be generated as such, the offline trained ANN shall select the online optimization algorithm which is capable of running in different conditions. Namely, the system comprises one classifier ANN structure and three optimization algorithm.

Altered optimization techniques shall come into effect relative to different day time periods and various conditions encountered. Classifier ANN engages the applicable optimization algorithm. Clock time and population number are the inputs to this system, and optimized average waiting time, total travel time and energy consumption are the outputs. The first stage of this thesis is performed by using MATLAB. MATLAB based elevator optimization is carried out as per obtained results. This study simulates 20-floor building with a 4-multicar system.

The thesis is organized as follows; chapter two provides elevator systems. Chapter three provides neural network and genetic algorithm, chapter four gives original work and simulations.

CHAPTER 2

2. Elevator Systems

An elevator system is a transportation mechanism that transfers passengers from one floor to another in a building. Elevators have been built throughout history but the first modern passenger elevators were developed no more than about 150 years ago. Steam and hydraulic elevators had already been introduced by 1852, when Elisha Otis made one of the most important elevator inventions, the clutch, which prevented the elevator from falling. Following this, in 1857, the first passenger elevator was installed in the store of E. Haughwout & Company, New York. With the advent of modern high-rise buildings, more elevator history than in any other single location was made in 1889, when the 321-meter-high Eiffel Tower was built for the Universal Exposition in Paris. In the Eiffel Tower, hydraulic double-deck elevators operated between ground level and the second platform. Between the second and third platforms two cars counterbalancing each other handled the traffic. The early hydraulic and steam-driven elevators functioned with pressurized water, which was either taken from the city water pipes or provided by steam engines. The elevator car was connected to a long piston that moved up when water was pumped into a cylinder, and came down when water was released by a hydraulic valve In 1880 Werner von Siemens introduced the utilization of electric power. Soon after, the geared or gearless traction electric elevators started to replace the hydraulic elevators. The development of electric elevators added impetus to high-rise construction. The fastest elevators today move at about 10 meters per second.

Elevators were first distributed by Strömberg. The Kone corporation was registered in 1910. In 1918 it started to design its own elevators, and the first four Kone elevators were delivered later that year. The first geared traction elevators from Kone were delivered to the Stockmann department store in 1928. Kone started to design its own AC and DC motors, and in the 1930's almost all the elevator components were being produced in-house. In electric elevators, the machinery for driving the elevator is usually located directly above the elevator hoistway in a machine room. After the middle of this century, hydraulic and ram elevators made a come-back, since it was cheaper to place the machine room at the bottom of the building. Modern hydraulic elevators function with oil, which is pumped by electric power. Since the speed of hydraulic elevators is low, they became popular especially in residential buildings, and in installations where heavy loads are transported. A recent innovation that requires no oil is a new hoisting unit where a synchronous axial motor is installed in the elevator shaft. With this concept the control boards are situated at the highest landing floor and no separate machine room is required. In a single elevator system the elevator control handles all the equipment. In the case of several elevators where each elevator is in its own shaft, the transportation capacity can be increased by a common group control that delivers the hall calls to the elevators. The first electrical controls were realized by relay techniques, the same kind of relays that were used in telephone exchanges. In the 1970's the principles of relay equipment were applied to electronic controls. In Kone, the first completely microprocessor-based control system was developed in the beginning of the 1980's,and it was first tested with a hybrid traffic simulator. With microprocessors, mathematical methods and sophisticated call allocation algorithms are used to optimize hall calls to the elevators. The rapid development of processor technology has made it possible to distribute the "intelligence" from the machine room to the elevator components, even to the hall call buttons. In a modern group control, a huge amount of information is handled during the hall call allocation compared to the old relay controls.

The average waiting times and queue lengths for a multi-car elevator system have been modelled with multi-bulk service queuing theory for the up-peak traffic. The up-peak situation, where passengers arrive at one entrance floor, i.e. at the lobby, and travel to the upper floors, is the most demanding for the elevator transportation capacity. In an up-peak situation the average time it takes for an elevator to serve the car calls and return back to the lobby, i.e. round trip time, is calculated using probability theory. In conventional elevator planning the up-peak handling capacity and interval are calculated from the round trip time value. The elevator group is chosen using standardized handling capacity and interval values for different types of buildings. In a real building, passengers arrive at several floors at the same time. Then the group control has a great effect on the passenger service times. The impact of the group control decisions on the passenger service times in different traffic situations cannot be calculated analytically. Passenger service times and elevator performance can be determined by simulating the contribution between passenger traffic flow and the elevator performance. Passengers arrive at an elevator system randomly and they are served in batches. The actual passenger arrival rates in a building can be assumed to follow the Poisson distribution. When a passenger arrives at a landing floor and gives a hall call, the group control allocates the call to the most suitable elevator. The time a hall call(landing call) stays on is called a hall call time. When a selected elevator starts to decelerate to the call floor, the hall call is cancelled. The system response time and call waiting time resemble the definition of the hall call time. Those times are measured from the moment of hall call registration until the car arrives to the floor and starts opening its doors.

In figure 2.1 refers, the passenger waiting time starts when a passenger arrives at a landing floor, and ends when he enters the elevator. The passenger ride time starts when a passenger enters a car and gives a car call (destination call), and ends when he exits the car at the destination floor. The total time a passenger spends within an elevator system first by waiting and then by riding inside a car is called the journey time, also called the service time and the time to destination. Journey time is approximated by average time to destination transit time to destination or by maximum passenger transit time to the destination. The travel time from the bottom to top floor at full speed is used to define the elevator speed. In elevator planning, a theoretical up-peak situation where all the elevators leave the lobby with 80 per-cent load is normally used. The two most important planning parameters, the up-peak interval and handling capacity, are calculated from the round trip time. The round trip time starts when a car opens its doors at the lobby, and continues until an elevator has made a trip around the building and starts to open its doors after returning to the lobby. The interval is the average time between car departures from the lobby. The handling capacity gives the number of passengers the elevator system can transport in five minutes during up-peak. [15]

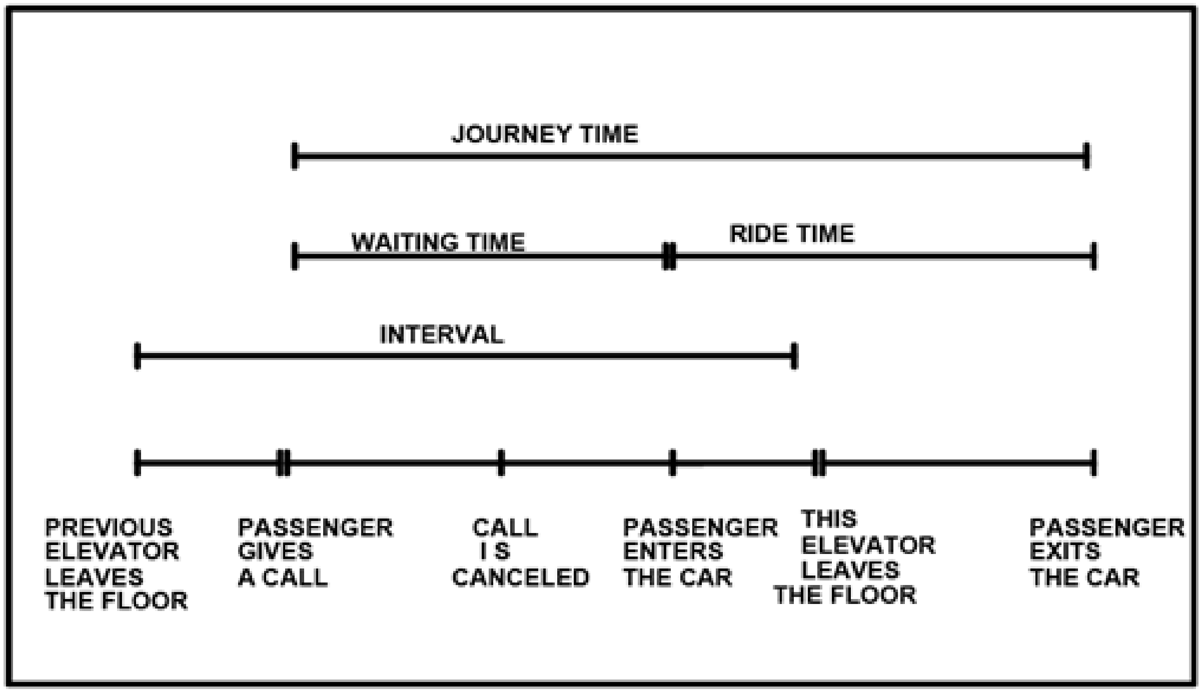

Figure 2.1: The Interval and the passenger service level parameters

The first elevators were operated by simple mechanical devices, such as "hand-rope" control. A passenger could call an elevator by operating a rope on both sides of the car. Since the shafts were not fully closed, the operation of elevators was quite unsafe. A primitive form of elevator control in a single car was based on an attendant-operated electrical car switch Using the switch, the attendant could manually drive the car up or down and decide at which floors to stop. The elevator efficiency and safety were increased with signaling devices at landings. Push-buttons were introduced in the 1920's to give the attendant information on the traffic demand, and the elevator shafts became closed. If no memory for the hall calls is provided, the calls are handled with a push-button control. With the non-collective controls the traffic demand is handled by serving each hall call at a time. A new hall call can be registered after the service of the previous call is completed. This control principle is used in freight elevators. When the registered calls are memorised, the elevator can pick several hall calls during the up or down trip. If there is only one call button at each floor, the calls can be arranged in a time queue according to the order they have been registered, or they can be served collectively. In the Interconnected Queue Selective (IQS) control system the hall calls are picked one at a time from the time queue so that the oldest call is served first. This type of control is used, for instance, in hospitals, where the bed calls are served one at a time. In collective control the car stops in floor sequence at each hall call. The Interconnected Down Collective (IDC) control system is often used in buildings where the traffic is mostly two-way between the ground level and the upper floors. This kind of traffic occurs, for example, in residential buildings. The elevator collects the hall calls during the down trip, i.e. serves the calls in sequence, stopping always at the nearest call floor. The IQS and IDC controls can be used by one car only or they can be applied for a group of elevators.

After automatic doors were developed in the 1950's, the traffic demand could be handled without attendants. In tall buildings, hall call buttons for both up and down directions were adapted. The most common call allocation principle, especially in the old relay controls, is the Interconnected Full Collective (IFC) control system. With two buttons at each floor, an elevator can pick the nearest hall call in front of the car in its direction of travel. The car calls given inside the elevator are always served in sequential order. After serving all the calls in the travel direction, the car moves to the furthest hall call in the opposite direction, where it reverses its direction. The efficiency of an elevator group was improved with a common central logic, the group control. The hall calls could be shared between several elevators where a common hall call button exists at every floor. The group control chooses the best elevator from a group of elevator cars to serve a given hall call. Group control dispatches cars to floors also for other reasons than hall calls, such as for parking, or if more than one elevator is needed at a busy floor. Elevators can be disconnected from the group for special service modes, such as emergency service, fireman's service or director/VIP service.

A disconnected elevator operates independently of the other elevators. One drawback of the collective control principle is the bunching of elevators. During heavy traffic there are a lot of hall calls to serve and the elevators have a tendency to move side by side, i.e. they start to bunch. This happens because elevators always stop at the nearest call and by-pass hall calls only when fully loaded. One of the early methods of preventing the bunching of elevators was to dispatch cars from the lobby at proper time intervals. A bus-type schedule for the elevators was applied. Elevators were delayed at the lobby for a certain time before they were sent to the upper floors. By delaying elevators at the lobby, part of the handling capacity was lost. In the 1970's collective control was first adapted to electronic controls. With electronic controls, however, collective controls were improved by giving priorities for the long or timed-out hall calls. Hall calls that had been on for a short time, were bypassed to get faster service for the timed-out calls. To some extent the elevators were kept apart from each other with this control. Peak traffic situations were handled with separate operation modes. In up-peak operation modes, such as next car up, dispatching intervals, zoning of floors, and later a channeling option were used.

The allocation principles of relay and electronic controls were brought to the microprocessor controls at the beginning of the 1980's. Hall calls were still prioritized according to the call service times. A slight but fundamental change was made in the philosophy. It was no longer expected that hall call times would become long before the "timed out" hall calls were given better service. The hall call times were forecast with mathematical calculations. When it was found that a hall call would become long with the normal service order, a car would bypass some hall calls to provide faster service to this call before it became long. When bypassing other hall calls, it was checked that the bypassed calls could be served by other cars within an acceptable time frame. On some occasions bonuses and penalties were used.

For example, a bonus was given to an elevator with a car call coinciding with the hall call when estimating the service time to the hall call. On the other hand, a penalty was given to some elevators, such as parked cars, when choosing the best car to serve a hall call. One important feature in the modern group supervisory controls is the time when hall calls are finally reserved to the cars. The reservation moment can be seen in the signalization at the landing call floor. As soon as the hall call is finally designated to a car, an arrow above the car-door opening is illuminated. Simultaneously an audible gong signal is given to inform the passenger which car is going to serve the given hall call. The final reservation must be stable not to mislead the passenger. To get the best optimization result, the reservation is often made at the latest possible moment, i.e. when the elevator starts to decelerate to the hall call floor. The other extreme is to reserve hall calls finally to an elevator immediately the hall call is given. This shortens the psychological waiting time of the passengers.

Passengers have more time to gather around the arriving elevator, which shortens the loading time. When the hall calls are allocated at an early stage, the future traffic events change the situation so that the early reservations are not as optimal as if the allocation was made at a later instant. In the course of the fast development of microprocessor technology, the call allocation algorithms have become more sophisticated. Mathematical methods are applied in the elevator controls. The traffic in the building is measured and learned in statistical forecasts. Statistical forecasts are used when allocating new hall calls to the cars. Statistical traffic forecasts are also adapted to the controls with a late hall call reservation, even though the allocation principle is not critical to the future traffic events. The methods that learn and adapt to the traffic of a building, and use rules based on expertise are connected to artificial intelligence in the elevator technology. The uncertainties in the predictions and control actions are described by fuzzy rules. Neural networks can make the elevator controls completely autonomous so that the control parameters are tuned by the traffic of the building. Then the significance of pre-defined expert rules diminishes

In the implementation of an elevator system, the following operating constraints must be met.

1. An elevator car must stop at all floors requested by the passengers traveling inside it.
2. An elevator car must not change its moving direction until all passengers traveling in the current direction have descended at their corresponding destination floors.

As lots of buildings with more than 40 floors are being built recently, so the elevator traffic control has become very important in the design of such high rise buildings. In general, one elevator group could serve up to 15–20 floors in the buildings depending on the building population. [4]

When a passenger arrives at a floor and gives a hall call, the system assigns the call to a suitable car. The passenger waiting time starts when a passenger arrives at a floor and presses a hall call button and it ends when the passenger enters into the car. Passenger traffic flow is conventionally classified into the following three patterns:

- Up-peak traffic: most passengers move up from the lobby to the upper floors and downward movements are rare (mostly in the early morning).
- Down-peak traffic: most passengers leave the floors and move down to the lobby and upward movements are rare (mostly in the evening).
- Regular traffic: passengers move up and down among several different floors (during the work time).

Up-peak and down-peak traffic are not simply equivalent in the sense of opposite directions. Up-peak traffic has a single arrival floor and many destinations, while down-peak traffic has many arrival floors and a single destination.

In an elevator system, many combinations of movements are possible. For an instance, in an elevator system with N floors, there are N-1 possible serviceable floors. an elevator from floor 1, can pass N-1 floors in respond to hall calls or car calls. Combining all the possible movements of an elevator from every floor, an elevator path for a building with 6 floors as shown at figure 2.2.

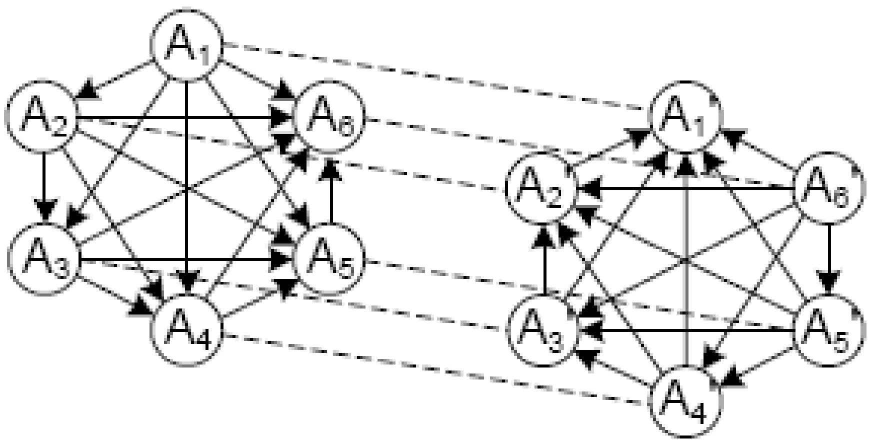

Figure2.2: Possible movements of elevator A from every floor.

In Figure 2.2, the path from A1 to A2 depicts the movement of elevator A from floor 1 to floor 2 (upward movement), and the path from A2' to A1' illustrates the movement from floor 2 to floor 1 (downward movement).

In high rise buildings, it is essential that the elevator systems comprise more than one elevator car for efficient transportation. For an elevator system that consists of three elevator cars, figure 2.3 can be obtained to demonstrate the paths.

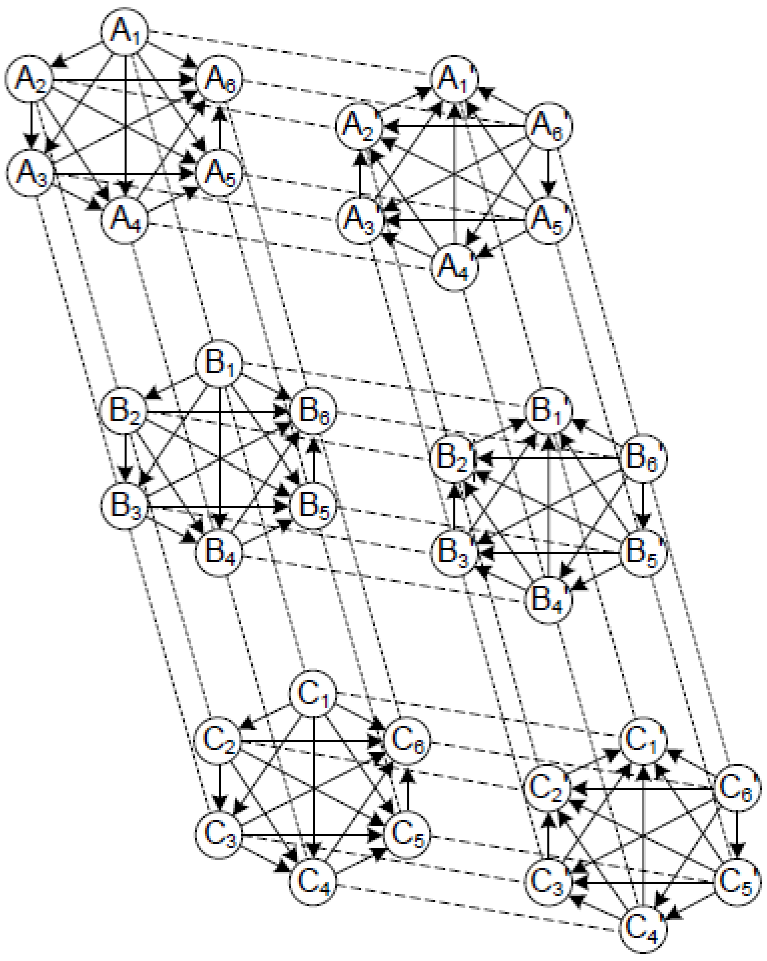

Figure 2.3: Combined possible paths on an elevator system with three elevators

The elevator group control system is a control system that manages systematically three of more elevators in a group to increase the service for passengers, and reduce the cost such as power consumption. Most of the elevator group control systems have used the hall call assignment method which assigns elevators in response to a passenger's call. The hall call assignment method assigns a new hall call to an elevator having the smallest evaluation function value among all the elevators.

Elevator group control systems are control systems that manage multiple elevators in a building in order to efficiently transport the passengers. The main requirements of an elevator group control system in serving both, car and hall calls are to provide even service to every floor in a building, to minimize the time spent by passengers waiting for service, to minimize the time spent by passengers to travel from one floor to another, to serve as many passengers as possible in a given time, to optimize power consumption, etc. Flow chart of the system is at figure 2.4 [19]

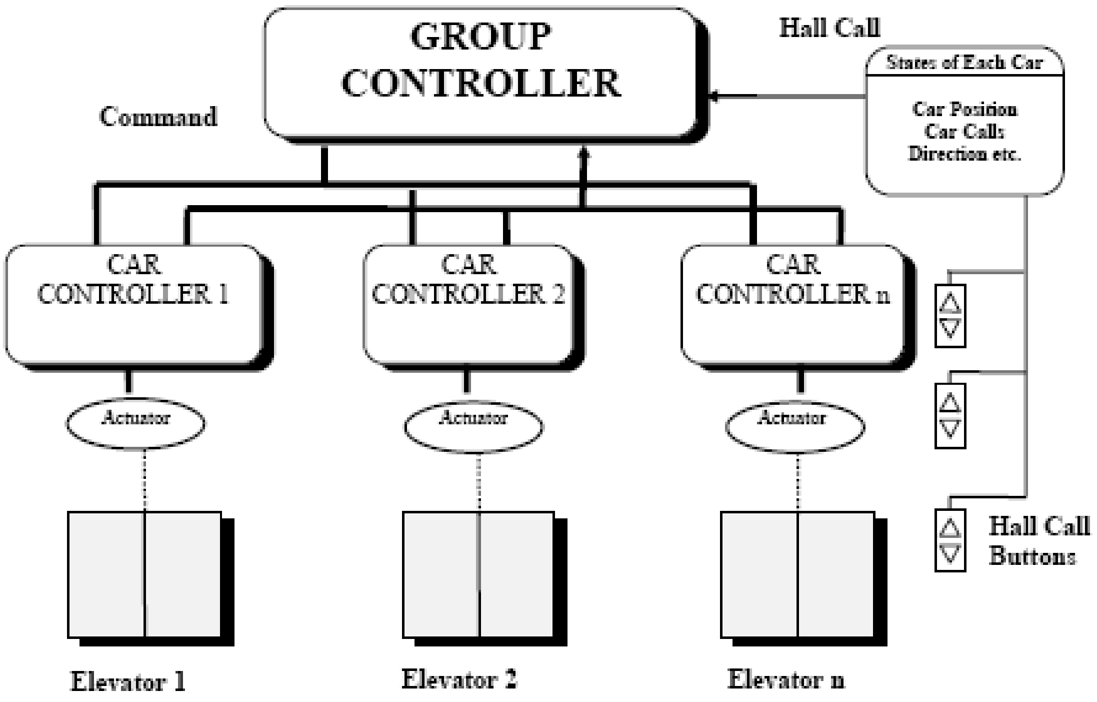

Figure 2.4: A general structure of an elevator group control system

Numerous conventional algorithms have been used to realize the elevator group controller which are listed as follow:

1. Hall call assignment method
2. Minimum long wait algorithm
3. Area-based control algorithm
4. Car-attribute based evaluation
5. Floor-attribute based evaluation

These conventional algorithms are based on evaluation functions which are calculated each time a hall call is made. An elevator group control system manages elevators so as to minimize the evaluation criteria; it is, however, difficult to satisfy all criteria or to take the actual situation of a building into account. Therefore, it is challenging for the elevator group controller to select a suitable elevator since an elevator system can very complex for the following reasons:

1. If a group controller manages n elevators and assigns p hall calls to the elevators, the controller considers np cases.
2. The controller must consider hall calls which will be generated in the near future.
3. It must consider many uncertain factors, such as number of passengers at the floors where hall calls and car calls are generated.
4. It must be possible for a system manager to change the control strategy. Some managers want to operate the system to minimize passengers' waiting time while others want to reduce the power consumption.

CHAPTER 3

3. ARTIFICIAL INTELLIGENCE AND GENETIC ALGORITHM

3.1 ARTIFICIAL NEURAL NETWORK SYSTEMS

The term neural network was traditionally used to refer to a network or circuit of biological neurons. The modern usage of the term often refers to artificial neural networks, which are composed of artificial neurons or nodes. Thus the term may refer to either biological neural networks are made up of real biological neurons or artificial neural networks for solving artificial intelligence problems.

Unlike von Neumann model computations, artificial neural networks do not separate memory and processing and operate via the flow of signals through the net connections, somewhat akin to biological networks.These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset.

A biological neural network is composed of a group or groups of chemically connected or functionally associated neurons. A single neuron may be connected to many other neurons and the total number of neurons and connections in a network may be extensive. Connections, called synapses, are usually formed from axons to dendrites, though dendrodendritic microcircuits and other connections are possible. Apart from the electrical signaling, there are other forms of signaling that arise from neurotransmitter diffusion.

Artificial intelligence, cognitive modelling, and neural networks are information processing paradigms inspired by the way biological neural systems process data. Artificial intelligence and cognitive modeling try to simulate some properties of biological neural networks. In the artificial intelligence field, artificial neural networks have been applied successfully to speech recognition, image analysis and adaptive control, in order to construct software agents (in computer and video games) or autonomous robots.

Historically, digital computers evolved from the von Neumann model, and operate via the execution of explicit instructions via access to memory by a number of processors. On the other hand, the origins of neural networks are based on efforts to model information processing in biological systems. Unlike the von Neumann model, neural network computing does not separate memory and processing.

Neural network theory has served both to better identify how the neurons in the brain function and to provide the basis for efforts to create artificial intelligence. The preliminary theoretical base for contemporary neural networks was independently proposed by Alexander Bain (1873) and William James (1890). In their work, both thoughts and body activity resulted from interactions among neurons within the brain.

For Bain, every activity led to the firing of a certain set of neurons. When activities were repeated, the connections between those neurons strengthened. According to his theory, this repetition was what led to the formation of memory. The general scientific community at the time was skeptical of Bain's theory because it required what appeared to be an inordinate number of neural connections within the brain. It is now apparent that the brain is exceedingly complex and that the same brain "wiring" can handle multiple problems and inputs.
James's theory was similar to Bain's, however, he suggested that memories and actions resulted from electrical currents flowing among the neurons in the brain. His model, by focusing on the flow of electrical currents, did not require individual neural connections for each memory or action.

C. S. Sherrington (1898) conducted experiments to test James's theory. He ran electrical currents down the spinal cords of rats. However, instead of demonstrating an increase in electrical current as projected by James, Sherrington found that the electrical current strength decreased as the testing continued over time. Importantly, this work led to the discovery of the concept of habituation.

McCulloch and Pitts (1943) created a computational model for neural networks based on mathematics and algorithms. They called this model threshold logic. The model paved the way for neural network research to split into two distinct approaches. One approach focused on biological processes in the brain and the other focused on the application of neural networks to artificial intelligence.

In the late 1940s psychologist Donald Hebb created a hypothesis of learning based on the mechanism of neural plasticity that is now known as Hebbian learning. Hebbian learning is considered to be a 'typical' unsupervised learning rule and its later variants were early models for long term potentiation. These ideas started being applied to computational models in 1948 with Turing's B-type machines.

Farley and Clark (1954) first used computational machines, then called calculators, to simulate a Hebbian network at MIT. Other neural network computational machines were created by Rochester, Holland, Habit, and Duda (1956).

Rosenblatt (1958) created the perceptron, an algorithm for pattern recognition based on a two-layer learning computer network using simple addition and subtraction. With mathematical notation, Rosenblatt also described circuitry not in the basic perceptron, such as the exclusive-or circuit, a circuit whose mathematical computation could not be processed until after the backpropagation algorithm was created by Werbos (1975).

Neural network research stagnated after the publication of machine learning research by Minsky and Papert (1969). They discovered two key issues with the computational machines that processed neural networks. The first issue was that single-layer neural networks were incapable of processing the exclusive-or circuit. The second significant issue was that computers were not sophisticated enough to effectively handle the long run time required by large neural networks. Neural network research slowed until computers achieved greater processing power. Also key in later advances was the backpropogation algorithm which effectively solved the exclusive-or problem (Werbos 1975).

The parallel distributed processing of the mid-1980s became popular under the name connectionism. The text by Rumelhart and McClelland (1986) provided a full exposition on the use of connectionism in computers to simulate neural processes. Neural networks, as used in artificial intelligence, have traditionally been viewed as simplified models of neural processing in the brain, even though the relation between this model and brain biological architecture is debated, as it is not clear to what degree artificial neural networks mirror brain function.

A neural network (NN), in the case of artificial neurons called artificial neural network (ANN) or simulated neural network (SNN), is an interconnected group of natural or artificial neurons that uses a mathematical or computational model for information processing based on a connectionistic approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network.

In more practical terms neural networks are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. However, the paradigm of neural networks - i.e., implicit, not explicit , learning is stressed - seems more to correspond to some kind of natural intelligence than to the traditional symbol-based Artificial Intelligence, which would stress, instead, rule-based learning.

An artificial neural network involves a network of simple processing elements (artificial neurons) which can exhibit complex global behavior, determined by the connections between the processing elements and element parameters. Artificial neurons were first proposed in 1943 by Warren McCulloch, a neurophysiologist, and Walter Pitts, a logician, who first collaborated at the University of Chicago. One classical type of artificial neural network is the recurrent Hopfield net.

In a neural network model simple nodes (which can be called by a number of names, including "neurons", "neurodes", "Processing Elements" (PE) and "units"), are connected together to form a network of nodes — hence the term "neural network". While a neural network does not have to be adaptive per se, its practical use comes with algorithms designed to alter the strength (weights) of the connections in the network to produce a desired signal flow.

The concept of a neural network appears to have first been proposed by Alan Turing in his 1948 paper Intelligent Machinery in which called them "B-type unorganised machines".

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations and also to use it. Unsupervised neural networks can also be used to learn representations of the input that capture the salient characteristics of the input distribution, e.g., see the Boltzmann machine (1983), and more recently, deep learning algorithms, which can implicitly learn the distribution function of the observed data. Learning in neural networks is particularly useful in applications where the complexity of the data or task makes the design of such functions by hand impractical. [20]

The tasks to which artificial neural networks are applied tend to fall within the following broad categories;

- Function approximation, or regression analysis, including time series prediction and modeling.

- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.

- Data processing, including filtering, clustering, blind signal separation and compression.

Application areas of ANNs include system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

Neuron consisits of three basic components; weights, thresholds and a single activation function. Neural network block diagram is at figure 3.1 [21]

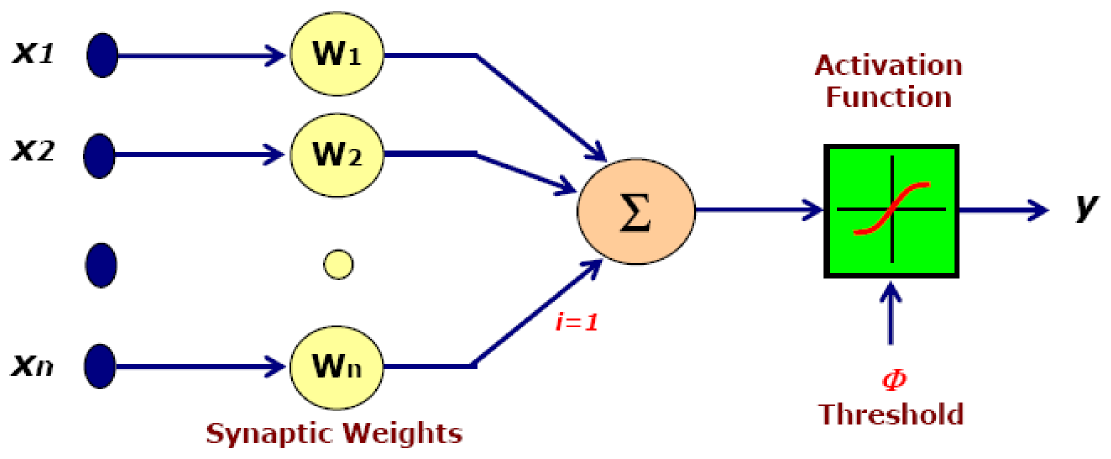

Figure 3.1: Neural Network Block Diagram

3.1.1 Type of Learning

3.1.1.1 Supervised Learning

In supervised training (Figure 3.2), both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights, which control the network. This process occurs over and over as the weights are continually tweaked. The set of data, which enables the training, is called the "training set." During the training of a network, the same set of data is processed many times, as the connection weights are ever refined. Sometimes a network may never learn. This could be because the input data does not contain the specific information from which the desired output is derived. Networks also don't converge if there is not enough data to enable complete learning. Ideally, there should be enough data so that part of the data can be held back as a test. Many layered networks with multiple nodes are capable of memorizing data. To monitor the network to determine if the system is simply memorizing its data in some non-significant way, supervised training needs to hold back a set of data to be used to test the system after it has undergone its training. If a network simply can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Another part of the designer's creativity governs the rules of training. There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is known as back-propagation. The training is not just a technique, but a conscious analysis, to insure that the network is not over trained. Initially, an artificial neural network configures itself with the general statistical trends of the data. Later, it continues to ‘learn’ about other aspects of the data, which may be spurious from a general viewpoint. When finally the system has been correctly trained and no further learning is needed, the weights can, if desired, be ‘frozen’. In some systems, this finalized network is then turned into hardware so that it can be fast. Other systems don't lock themselves in but continue to learn while in production use.

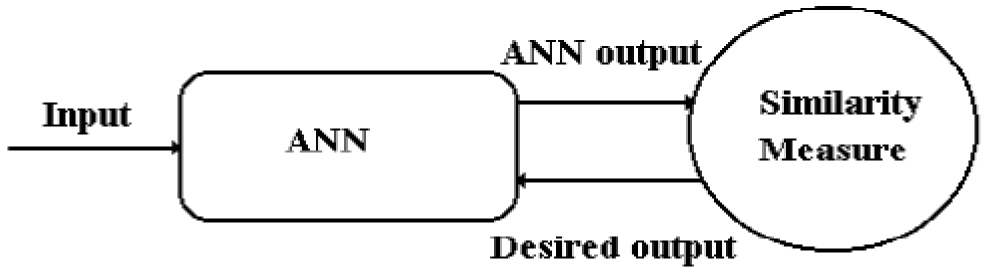

Figure 3.2: Supervised Learning

3.1.1.2 Reinforcement Learning

This type of learning may be considered as an intermediate form of the above two types of learning. Here the learning machine does some action on the environment and gets a feedback response from the environment. The learning system grades its action good (rewarding) or bad (punishable) based on the environmental response and accordingly adjusts its parameters. Generally, parameter adjustment is continued until an equilibrium state occurs, following which there will be no more changes in its parameters. The selforganizing neural learning (Figure 3.3) may be categorized under this type of learning.

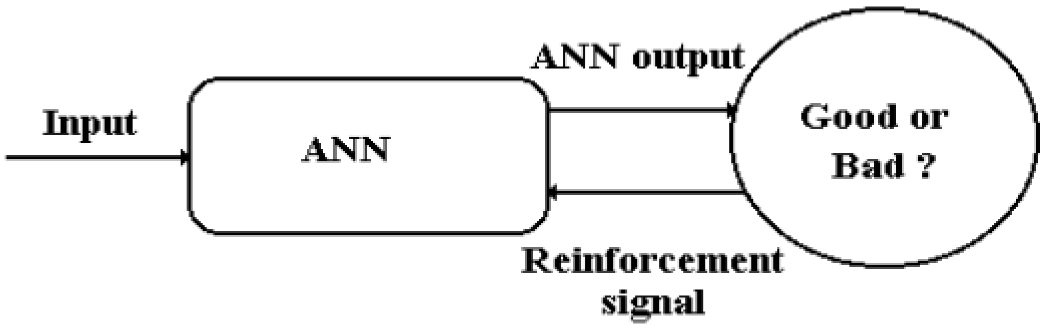

Figure 3.3: Reinforcement Learning

3.1.1.3 Unsupervised Learning

The other type is the unsupervised training (learning) (Figure 3.4). In this type, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption. These networks use no external influences to adjust their weights. Instead, they internally monitor their performance. These networks look for regularities or trends in the input signals, and makes adaptations according to the function of the network. Even without being told whether it's right or wrong, the network still must have some information about how to organize itself. This information is built into the network topology and learning rules. An unsupervised learning algorithm might emphasize cooperation among clusters of processing elements. In such a scheme, the clusters would work together. If some external input activated any node in the cluster, the cluster's activity as a whole could be increased. Likewise, if external input to nodes in the cluster was decreased, that could have an inhibitory effect on the entire cluster. Competition between processing elements could also form a basis for learning. Training of competitive clusters could amplify the responses of specific groups to specific stimuli. As such, it would associate those groups with each other and with a specific appropriate response. Normally, when competition for learning is in effect, only the weights belonging to the winning processing element will be updated. Presently, the unsupervised learning is not well understood and there continues to be a lot of research in this aspect.

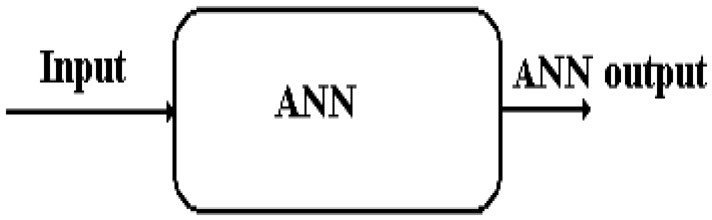

Figure 3.4: Unsupervised Learning

3.1.2 TYPES OF ARTIFICIAL NEURAL NETWORKS

3.1.2.1 SINGLE-LAYER FEED FORWARD NETWORK

A neural network in which the input layer of source nodes projects into an output layer of neurons but not vice-versa is known as single feed-forward or acyclic network. In single layer network, ‘single layer’ refers to the output layer of computation nodes as shown at figure 3.5.

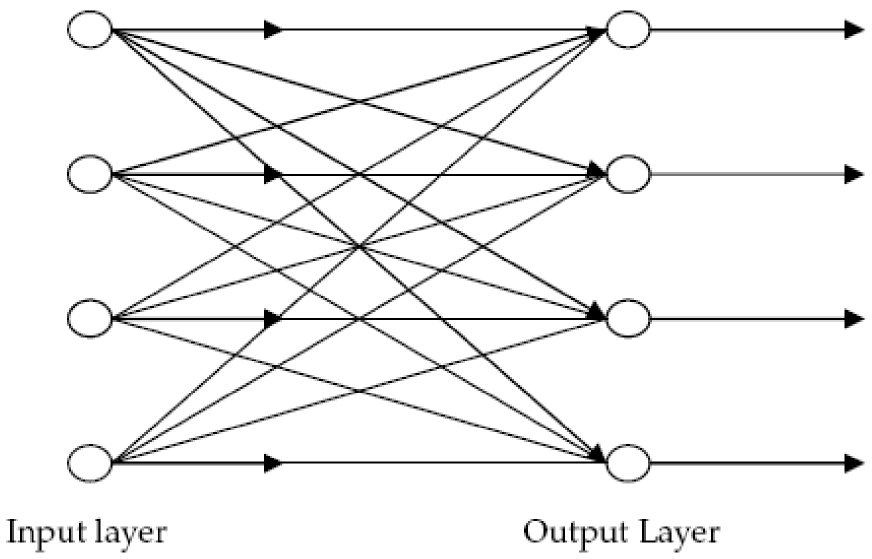

Figure 3.5: A Single Layer-Feedforward Network

3.1.2.2 MULTI-LAYER FEED FORWARD NETWORK

This type of network consists of one or more hidden layers, whose computation nodes are called hidden neurons or hidden units as shown at figure 3.6. The function of hidden neurons is to interact between the external input and network output in some useful manner and to extract higher order statistics. The source nodes in input layer of network supply the input signal to neurons in the second layer (First hidden layer). The output signals of second. layer are used as inputs to the third layer and so on. The set of output signals of the neurons in the output layer of network constitutes the overall response of network to the activation pattern supplied by source nodes in the input first layer.

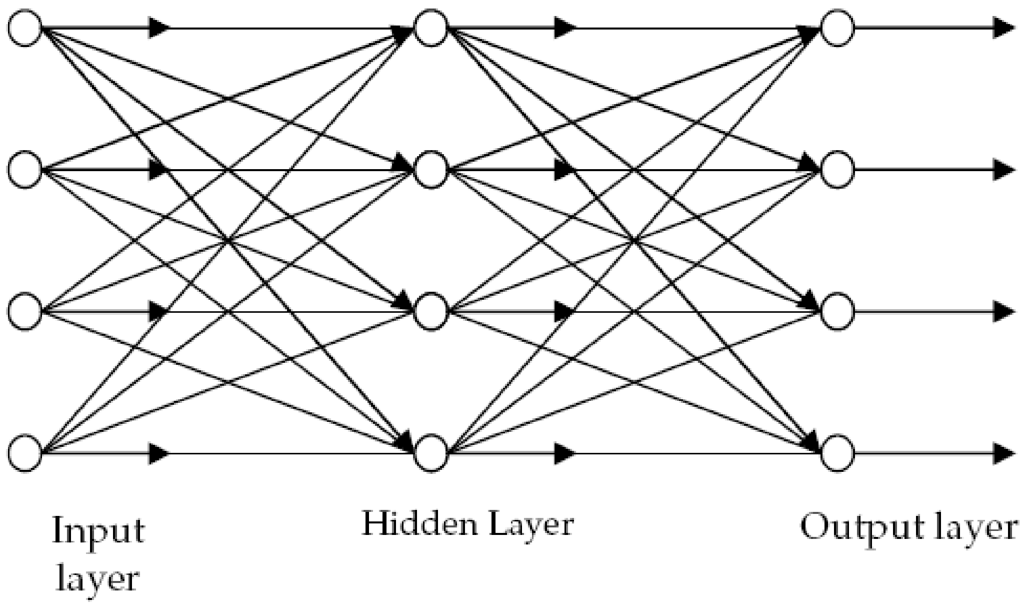

Figure 3.6: A Multi Layer-Feedforward Network

Short characterization of feedforward networks

1. Typically, activation is fed forward from input to output through ‘hidden layers’, though many other architectures exist.
2. Mathematically, they implement static input-output mappings.
3. Most popular supervised training algorithm: backpropagation algorithm
4. Have proven useful in many practical applications as approximators of nonlinear functions and as pattern classificators.

3.2 GENETIC ALGORITHM

Computer simulations of evolution started as early as in 1954 with the work of Nils Aall Barricelli, who was using the computer at the Institute for Advanced Study in Princeton, New Jersey. His 1954 publication was not widely noticed. Starting in 1957, the Australian quantitative geneticist Alex Fraser published a series of papers on simulation of artificial selection of organisms with multiple loci controlling a measurable trait. From these beginnings, computer simulation of evolution by biologists became more common in the early 1960s, and the methods were described in books by Fraser and Burnell (1970) and Crosby (1973). Fraser's simulations included all of the essential elements of modern genetic algorithms. In addition, Hans-Joachim Bremermann published a series of papers in the 1960s that also adopted a population of solution to optimization problems, undergoing recombination, mutation, and selection. Bremermann's research also included the elements of modern genetic algorithms. Other noteworthy early pioneers include Richard Friedberg, George Friedman, and Michael Conrad. Many early papers are reprinted by Fogel (1998).

Although Barricelli, in work he reported in 1963, had simulated the evolution of ability to play a simple game, artificial evolution became a widely recognized optimization method as a result of the work of Ingo Rechenberg and Hans-Paul Schwefel in the 1960s and early 1970s – Rechenberg's group was able to solve complex engineering problems through evolution strategies. Another approach was the evolutionary programming technique of Lawrence J. Fogel, which was proposed for generating artificial intelligence. Evolutionary programming originally used finite state machines for predicting environments, and used variation and selection to optimize the predictive logics. Genetic algorithms in particular became popular through the work of John Holland in the early 1970s, and particularly his book Adaptation in Natural and Artificial Systems (1975). His work originated with studies of cellular automata, conducted by Holland and his students at the University of Michigan. Holland introduced a formalized framework for predicting the quality of the next generation, known as Holland's Schema Theorem. Research in GAs remained largely theoretical until the mid-1980s, when The First International Conference on Genetic Algorithms was held in Pittsburgh, Pennsylvania.

As academic interest grew, the dramatic increase in desktop computational power allowed for practical application of the new technique. In the late 1980s, General Electric started selling the world's first genetic algorithm product, a mainframe-based toolkit designed for industrial processes. In 1989, Axcelis, Inc. released Evolver, the world's first commercial GA product for desktop computers. The New York Times technology writer John Markoff wrote about Evolver in 1990.

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics (Fraser, 1957; Bremermann, 1958; Holland, 1975). GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles. For example, in a problem such as the traveling salesman problem, a chromosome represents a route, and a gene may represent a city. In contrast to traditional optimization techniques, GAs work with coding of parameters, rather than the parameters themselves. To evolve good solutions and to implement natural selection, we need ameasure for distinguishing good solutions from bad solutions. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions over worse ones. In essence, the fitness measure must determine a candidate solution’s relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions. Another important concept of GAs is the notion of population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms. For example, small population sizes might lead to prematüre. A genetic algorithm is a probabilistic search technique that computationally simulates the process of biological evolution. It mimics evolution in nature by repeatedly altering a population of candidate solutions until an optimal solution is found.

The GA evolutionary cycle starts with a randomly selected initial population. The changes to the population occur through the processes of selection based on fitness, and alteration using crossover and mutation. The application of selection and alteration leads to a population with a higher proportion of better solutions. The evolutionary cycle (at figure 3.7) continues until an acceptable solution is found in the current generation of population, or some control parameter such as the number of generations is exceeded.

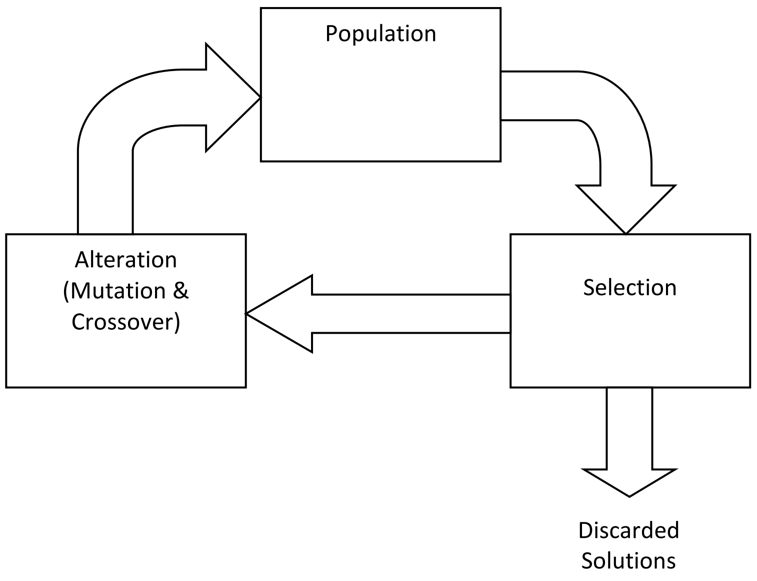

Figure 3.7: Genetic Algorithm Evolutionary Cycle.

The smallest unit of a genetic algorithm is called a gene, which represents a unit of information in the problem domain. A series of genes, known as a chromosome, represents one possible solution to the problem. Each gene in the chromosome represents one component of the solution pattern.

The most common form of representing a solution as a chromosome is a string of binary digits. Each bit in this string is a gene. The process of converting the solution from its original form into the bit string is known as coding. The specific coding scheme used is application dependent. The solution bit strings are decoded to enable their evaluation using a fitness measure. [21]

3.2.1 Methodology

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

A typical genetic algorithm requires:

1. A genetic representation of the solution domain,
2. A fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

3.2.2 Selection

In biological evolution, only the fittest survive and their gene pool contributes to the creation of the next generation. Selection in GA is also based on a similar process. In a common form of selection, known as fitness proportional selection, each chromosome's likelihood of being selected as a good one is proportional to its fitness value.

3.2.2.1 Fitness Proportionate Selection

This includes methods such as roulette-wheel selection (Holland, 1975; Goldberg, 1989b) and stochastic universal selection (Baker, 1985; Grefenstette and Baker, 1989). In roulette-wheel selection, each individual in the population is assigned a roulette. Wheel slot sized in proportion to its fitness. That is, in the biased roulette wheel, good solutions have a larger slot size than the less fit solutions. The roulette wheel is spun to obtain a reproduction candidate. The roulettewheel selection scheme can be implemented as follows:

1. Evaluate the fitness, fi, of each individual in the population.
2. Compute the probability (slot size), pi, of selecting each member of the population:
3. Calculate the cumulative probability, qi , for each individual
4. Generate a uniform random number, $r \in (0, 1]$.
5. If $r < q1$ then select the first chromosome, x1, else select the individual xi such that $qi-1 < r \leq qi$.
6. Repeat steps 4–5 n times to create n candidates in the mating pool.

3.2.2.2 Ordinal Selection

This includes methods such as tournament selection (Goldberg et al., 1989b), and truncation selection (M¨uhlenbein and Schlierkamp-Voosen, 1993). In tournament selection, s chromosomes are chosen at random (either with or without replacement) and entered into a tournament against each other. The fittest individual in the group of k chromosomes wins the tournament and is selected as the parent. The most widely used value of s is 2. Using this selection scheme, n tournaments are required to choose n individuals. In truncation selection, the top (1/s)th of the individuals get s copies each in the mating pool.

3.2.2.3 Alteration to improve good solutions

The alteration step in the genetic algorithm refines the good solution from the current generation to produce the next generation of candidate solutions. It is carried out by performing crossover and mutation.

3.2.3 Crossover

Crossover (figure 3.8) may be regarded as artificial mating in which chromosomes from two individuals are combined to create the chromosome for the next generation. This is done by splicing two chromosomes from two different solutions at a crossover point and swapping the spliced parts. The idea is that some genes with good characteristics from one chromosome may as a result combine with some good genes in the other chromosome to create a better solution represented by the new chromosome.

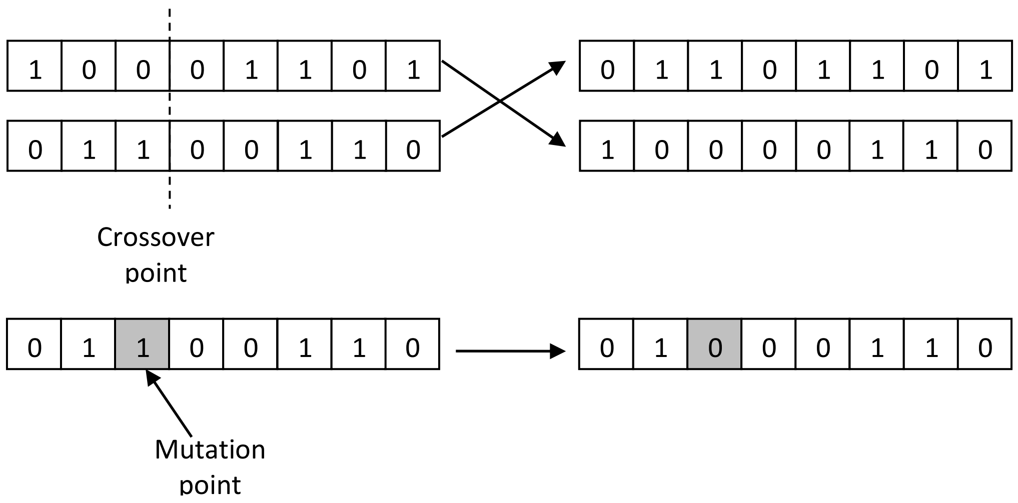

Figure 3.8: Crossover & Mutation

3.2.3.1 K-Point Crossover

One-point, and two-point crossovers are the simplest and most widely applied crossover methods. In one-point crossover, , a crossover site is selected at random over the string length, and the alleles on one side of the site are exchanged between the individuals. In two-point crossover, two crossover sites are randomly selected. The alleles between the two sites are exchanged between the two randomly paired individuals.. The concept of one-point crossover can be extended to k-point crossover, where k crossover points are used, rather than just one or two.

3.2.3.2 Uniform Crossover

Another common recombination operator is uniform crossover (Syswerda, 1989; Spears and De Jong, 1994). In uniform crossover, every allele is exchanged between the a pair of randomly selected chromosomes with a certain probability, pe, known as the swapping probability. Usually the swapping probability value is taken to be 0.5.

3.2.3.3 Uniform Order-Based Crossover

The k-point and uniform crossover (Figure 3.9) methods described above are not well suited for search problems with permutation codes such as the ones used in the traveling salesman problem. They often create offspring that represent invalid solutions for the search problem.

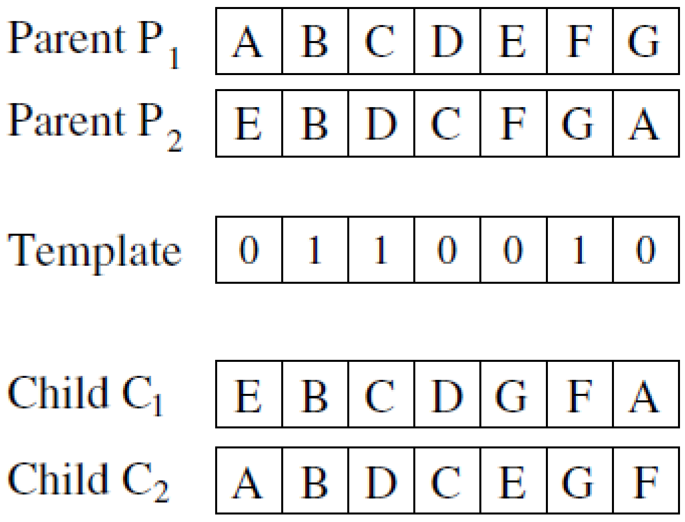

Figure 3.9: Illustration of Uniform order crossover

3.2.3.4 Order-Based Crossover

The order-based crossover operator (Davis, 1985) at figure 3.10 and 3.11 is a variation of the uniform order-based crossover in which two parents are randomly selected and two random crossover sites are generated. The genes between the cut points are copied to the children. Starting from the second crossover site copy the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear. for offspring C1, since alleles C, D, and E are copied from the parent P1, we get alleles B, G, F, and A from the parent P2. Starting from the second crossover site, which is the sixth gene, we copy alleles B and G as the sixth and seventh genes respectively. We then wrap around and copy alleles F and A as the first and second genes.

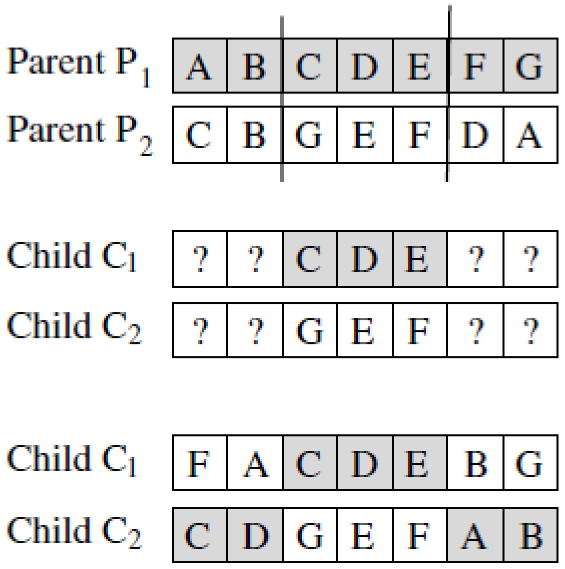

Figure 3.10: Illustration of Order-Based crossover

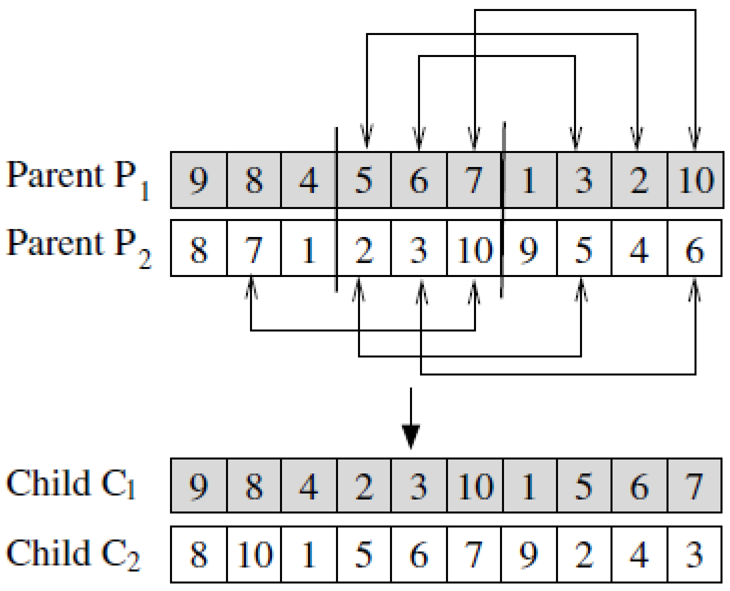

Figure 3.11: Illustration of Partially Matched Crossover

3.2.3.5 Partially Matched Crossover (PMX)

Apart from always generating valid offspring, the PMX operator (Goldberg and Lingle, 1985) also preserves orderings within the chromosome. In PMX, two parents are randomly selected and two random crossover sites are generated. Alleles within the two crossover sites of a parent are exchanged with the alleles corresponding to those mapped by the other parent. For example at figure 3.12 looking at parent P1, the first gene within the two crossover sites, 5, maps to 2 in P2. Therefore, genes 5 and 2 are swapped in P1. Similarly we swap 6 and 3, and 10 and 7 to create the offspring C1. After all exchanges it can be seen that we have achieved a duplication of the ordering of one of the genes in between the crossover point within the opposite chromosome, and vice versa.

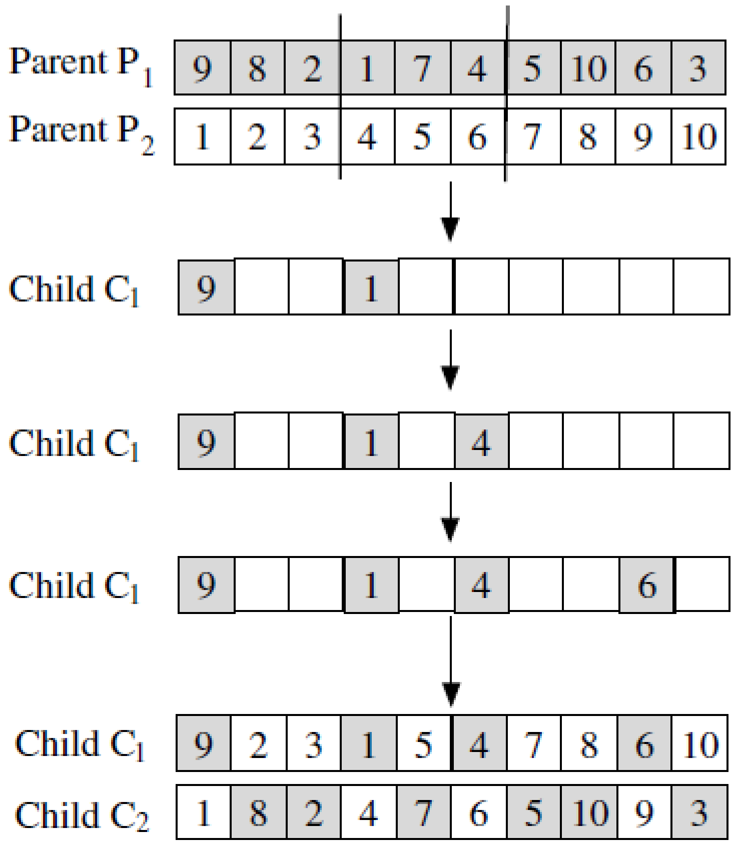

Figure 3.12: Illustration of Cycle Crossover

3.2.4 Mutation

Mutation is a random adjustment in the genetic composition. It is useful for introducing new characteristics in a population – something not achieved through crossover alone. Crossover only rearranges existing characteristics to give new combinations. For example, if the first bit in every chromosome of a generation happens to be a 1, any new chromosome created through crossover will also have 1 as the first bit.

The mutation operator changes the current value of a gene to a different one. For bit string chromosome this change amounts to flipping a 0 bit to a 1 or vice versa.

Although useful for introducing new traits in the solution pool, mutations can be counterproductive, and applied only infrequently and randomly.

The steps in the typical genetic algorithm for finding a solution to a problem are listed below:

1. Create an initial solution population of a certain size randomly

2. Evaluate each solution in the current generation and assign it a fitness value.

3. Select "good" solutions based on fitness value and discard the rest.

4. If acceptable solution(s) found in the current generation or maximum number of generations is exceeded then stop.

5. Alter the solution population using crossover and mutation to create a new generation of solutions.

6. Go to step 2.

3.2.5 Limitations

There are several limitations of the use of a genetic algorithm compared to alternative optimization algorithms:

- Repeated fitness function evaluation for complex problems is often the most prohibitive and limiting segment of artificial evolutionary algorithms. Finding the optimal solution to complex high dimensional, multimodal problems often requires very expensive fitness function evaluations. In real world problems such as structural optimization problems, one single function evaluation may require several hours to several days of complete simulation. Typical optimization methods can not deal with such types of problem. In this case, it may be necessary to forgo an exact evaluation and use an approximated fitness that is computationally efficient. It is apparent that amalgamation of approximate models may be one of the most promising approaches to convincingly use GA to solve complex real life problems.

- Genetic algorithms do not scale well with complexity. That is, where the number of elements which are exposed to mutation is large there is often an exponential increase in search space size. This makes it extremely difficult to use the technique on problems such as designing an engine, a house or plane. In order to make such problems tractable to evolutionary search, they must be broken down into the simplest representation possible. Hence we typically see evolutionary algorithms encoding designs for fan blades instead of engines, building shapes instead of detailed construction plans, airfoils instead of whole aircraft designs. The second problem of complexity is the issue of how to protect parts that have evolved to represent good solutions from further destructive mutation, particularly when their fitness assessment requires them to combine well with other parts. It has been suggested by some[citation needed] in the community that a developmental approach to evolved solutions could overcome some of the issues of protection, but this remains an open research question.

- The "better" solution is only in comparison to other solutions. As a result, the stop criterion is not clear in every problem.

- In many problems, GAs may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. This means that it does not "know how" to sacrifice short-term fitness to gain longer-term fitness. The likelihood of this occurring depends on the shape of the fitness landscape: certain problems may provide an easy ascent towards a global optimum, others may make it easier for the function to find the local optima. This problem may be alleviated by using a different fitness function, increasing the rate of mutation, or by using selection techniques that maintain a diverse population of solutions, although the No Free Lunch theorem proves that there is no general solution to this problem. A common technique to maintain diversity is to impose a "niche penalty", wherein, any group of individuals of sufficient similarity (niche radius) have a penalty added, which will reduce the representation of that group in subsequent generations, permitting other (less similar) individuals to be maintained in the population. This trick, however, may not be effective, depending on the landscape of the problem. Another possible technique would be to simply replace part of the population with randomly generated individuals, when most of the population is too similar to each other. Diversity is important in genetic algorithms (and genetic programming) because crossing over a homogeneous population does not yield new solutions. In evolution strategies and evolutionary programming, diversity is not essential because of a greater reliance on mutation.

- Operating on dynamic data sets is difficult, as genomes begin to converge early on towards solutions which may no longer be valid for later data. Several methods have been proposed to remedy this by increasing genetic diversity somehow and preventing early convergence, either by increasing the probability of mutation when the solution quality drops (called triggered hypermutation), or by occasionally introducing entirely new, randomly generated elements into the gene pool (called random immigrants). Again, evolution strategies and evolutionary programming can be implemented with a so-called "comma strategy" in which parents are not maintained and new parents are selected only from offspring. This can be more effective on dynamic problems.

- GAs cannot effectively solve problems in which the only fitness measure is a single right/wrong measure (like decision problems), as there is no way to converge on the solution (no hill to climb). In these cases, a random search may find a solution as quickly as a GA. However, if the situation allows the success/failure trial to be repeated giving (possibly) different results, then the ratio of successes to failures provides a suitable fitness measure.

- For specific optimization problems and problem instances, other optimization algorithms may find better solutions than genetic algorithms (given the same amount of computation time). Alternative and complementary algorithms include evolution strategies, evolutionary programming, simulated annealing, Gaussian adaptation, hill climbing, and swarm intelligence (e.g.: ant colony optimization, particle swarm optimization) and methods based on integer linear programming. The question of which, if any, problems are suited to genetic algorithms (in the sense that such algorithms are better than others) is open and controversial.

CHAPTER 4

4. APPLICATION OF HYBRID METHOD TO GROUP ELEVATOR CONTROL

The Elevator Group Controller is a control system that accomplishes the systematic management of two or more elevators so that the elevator system is operated at best balance of energy, time and demand. Neural Network and Artificial Intelligence presents convenient options in designing a controller that capable of solving aforementioned numerous objectives. However, since in this particular problem, multiple inputs and and outputs have to be considered, complications might arise in forming neural networks and genetic algorithms. Remedy to this problem is that, ordinal structured neural network shall classify the inputs and then allocates these inputs to related genetic algoritm codes to optimize the selected variables. Block diagram showed as figure 4.1;

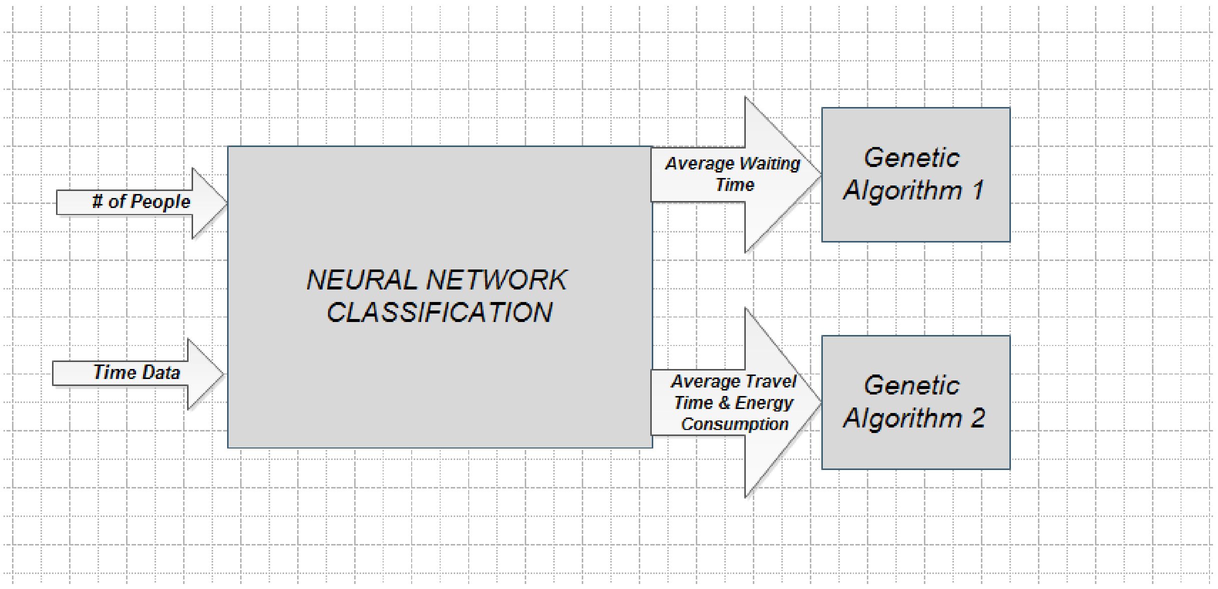

Figure 4.1: Block Diagram of the Suggested Method

4.1 ANN FOR THE SELECTION OF OPTIMIZATION ALGORIHTM

In this project; we used Neural Network for classificate the inputs to select one of the optimization algorithm. These inputs are the human population and time data.

Input Parameters;

- *Time (D1)*
    - From 00:00 to 23:59 (Whole 24 hours)
        - 00:00 – 07:59 -> Less Traffic
        - 08:00 – 08:59 -> Morning Peak
        - 09:00 – 11:59 -> Less Traffic
        - 12:00 – 13:29 -> Lunch Peak
        - 13:30 – 17:59 ->Less Traffic
        - 18:00 – 18:29 ->Evening Peak
        - 18:30 – 23:59 -> Less Traffic
    - With 5 minutes intermittently (Own coefficients 0.1 intermittently)
- *Crowdedness (D2)*
    - 0-7 people called **“Solitary” (Coefficent: 0.3)**
    - 7-11 people called **“Middle Crowded” (Coefficent: 0.5)**
    - 11-16 people called **“Overcrowded” (Coefficent: 0.3)**

Output Parameters;

- *Passenger Average Waiting Time (A1)*
- *Passenger Travel Time and Energy Consumption (A2)*

And some examples from a 864 sample neural network table as;

INPUT PARAMETERS				**OUTPUT PARAMETERS**	
Coefficient of D1	***D1***	***Coefficient of D2***	***D2***	***A1***	***A2***
0,1	00:00 - 00:04	0.5	Solitary	0	1
0,2	00:05 - 00:09	0.5	Solitary	0	1
0,3	00:10 - 00:14	0.5	Solitary	0	1
0,4	00:15 - 00:19	0.5	Solitary	0	1
0,5	00:20 - 00:24	0.5	Solitary	0	1
0,6	00:25 - 00:29	0.5	Solitary	0	1
0,7	00:30 - 00:34	0.5	Solitary	0	1
0,8	00:35 - 00:39	0.5	Solitary	0	1
0,9	00:40 - 00:44	0.5	Solitary	0	1
1	00:45 - 00:49	0.5	Solitary	0	1
1,1	00:50 - 00:54	0.5	Solitary	0	1
1,2	00:55 - 00:59	0.5	Solitary	0	1
1,3	01:00 - 01:04	0.5	Solitary	0	1

1,4	01:05 - 00:09	0.5	Solitary	0	1
1,5	01:10 - 01:14	0.5	Solitary	0	1
1,6	01:15 - 01:19	0.5	Solitary	0	1
1,7	01:20 - 01:24	0.5	Solitary	0	1
1,8	01:25 - 01:29	0.5	Solitary	0	1
1,9	01:30 - 01:34	0.5	Solitary	0	1
2	01:35 - 01:39	0.5	Solitary	0	1
2,1	01:40 - 01:44	0.5	Solitary	0	1
2,2	01:45 - 01:49	0.5	Solitary	0	1
2,3	01:50 - 01:54	0.5	Solitary	0	1
2,4	01:55 - 01:59	0.5	Solitary	0	1
2,5	02:00 - 02:04	0.5	Solitary	0	1
2,6	02:05 - 02:09	0.5	Solitary	0	1
2,7	02:10 - 02:14	0.5	Solitary	0	1
2,8	02:15 - 02:19	0.5	Solitary	0	1
2,9	02:20 - 02:24	0.5	Solitary	0	1
3	02:25 - 02:29	0.5	Solitary	0	1
3,1	02:30 - 02:34	0.5	Solitary	0	1
3,2	02:35 - 02:39	0.5	Solitary	0	1
3,3	02:40 - 02:44	0.5	Solitary	0	1
3,4	02:45 - 02:49	0.5	Solitary	0	1
3,5	02:50 - 02:54	0.5	Solitary	0	1
3,6	02:55 - 02:59	0.5	Solitary	0	1
3,7	03:00 - 03:04	0.5	Solitary	0	1
3,8	03:05 - 03:09	0.5	Solitary	0	1
3,9	03:10 - 03:14	0.5	Solitary	0	1
4	03:15 - 03:19	0.5	Solitary	0	1
4,1	03:20 - 03:24	0.5	Solitary	0	1
4,2	03:25 - 03:29	0.5	Solitary	0	1
4,3	03:30 - 03:34	0.5	Solitary	0	1
4,4	03:35 - 03:39	0.5	Solitary	0	1
4,5	03:40 - 03:44	0.5	Solitary	0	1
4,6	03:45 - 03:49	0.5	Solitary	0	1
4,7	03:50 - 03:54	0.5	Solitary	0	1
4,8	03:55 - 03:59	0.5	Solitary	0	1
4,9	04:00 - 04:04	0.5	Solitary	0	1
5	04:05 - 04:09	0.5	Solitary	0	1
5,1	04:10 - 04:14	0.5	Solitary	0	1
5,2	04:15 - 04:19	0.5	Solitary	0	1
5,3	04:20 - 04:24	0.5	Solitary	0	1
5,4	04:25 - 04:29	0.5	Solitary	0	1
5,5	04:30 - 04:34	0.5	Solitary	0	1
5,6	04:35 - 04:39	0.5	Solitary	0	1
….	…….	…..	……	…..	…..
0,1	00:00 - 00:04	0.5	Middle Crowded	0	1

0,2	00:05 - 00:09	0.5	Middle Crowded	0	1
0,3	00:10 - 00:14	0.5	Middle Crowded	0	1
0,4	00:15 - 00:19	0.5	Middle Crowded	0	1
0,5	00:20 - 00:24	0.5	Middle Crowded	0	1
0,6	00:25 - 00:29	0.5	Middle Crowded	0	1
0,7	00:30 - 00:34	0.5	Middle Crowded	0	1
0,8	00:35 - 00:39	0.5	Middle Crowded	0	1
0,9	00:40 - 00:44	0.5	Middle Crowded	0	1
1	00:45 - 00:49	0.5	Middle Crowded	0	1
1,1	00:50 - 00:54	0.5	Middle Crowded	0	1
1,2	00:55 - 00:59	0.5	Middle Crowded	0	1
1,3	01:00 - 01:04	0.5	Middle Crowded	0	1
1,4	01:05 - 00:09	0.5	Middle Crowded	0	1
1,5	01:10 - 01:14	0.5	Middle Crowded	0	1
1,6	01:15 - 01:19	0.5	Middle Crowded	0	1
1,7	01:20 - 01:24	0.5	Middle Crowded	0	1
1,8	01:25 - 01:29	0.5	Middle Crowded	0	1
1,9	01:30 - 01:34	0.5	Middle Crowded	0	1
2	01:35 - 01:39	0.5	Middle Crowded	0	1
2,1	01:40 - 01:44	0.5	Middle Crowded	0	1
2,2	01:45 - 01:49	0.5	Middle Crowded	0	1
2,3	01:50 - 01:54	0.5	Middle Crowded	0	1
2,4	01:55 - 01:59	0.5	Middle Crowded	0	1
2,5	02:00 - 02:04	0.5	Middle Crowded	0	1
2,6	02:05 - 02:09	0.5	Middle Crowded	0	1
2,7	02:10 - 02:14	0.5	Middle Crowded	0	1
2,8	02:15 - 02:19	0.5	Middle Crowded	0	1
2,9	02:20 - 02:24	0.5	Middle Crowded	0	1
3	02:25 - 02:29	0.5	Middle Crowded	0	1
3,1	02:30 - 02:34	0.5	Middle Crowded	0	1
3,2	02:35 - 02:39	0.5	Middle Crowded	0	1
3,3	02:40 - 02:44	0.5	Middle Crowded	0	1
3,4	02:45 - 02:49	0.5	Middle Crowded	0	1
3,5	02:50 - 02:54	0.5	Middle Crowded	0	1
3,6	02:55 - 02:59	0.5	Middle Crowded	0	1
3,7	03:00 - 03:04	0.5	Middle Crowded	0	1
3,8	03:05 - 03:09	0.5	Middle Crowded	0	1
3,9	03:10 - 03:14	0.5	Middle Crowded	0	1
4	03:15 - 03:19	0.5	Middle Crowded	0	1
4,1	03:20 - 03:24	0.5	Middle Crowded	0	1
4,2	03:25 - 03:29	0.5	Middle Crowded	0	1
4,3	03:30 - 03:34	0.5	Middle Crowded	0	1
4,4	03:35 - 03:39	0.5	Middle Crowded	0	1
4,5	03:40 - 03:44	0.5	Middle Crowded	0	1
4,6	03:45 - 03:49	0.5	Middle Crowded	0	1

4,7	03:50 - 03:54	0.5	Middle Crowded	0	1
4,8	03:55 - 03:59	0.5	Middle Crowded	0	1
4,9	04:00 - 04:04	0.5	Middle Crowded	0	1
5	04:05 - 04:09	0.5	Middle Crowded	0	1
5,1	04:10 - 04:14	0.5	Middle Crowded	0	1
5,2	04:15 - 04:19	0.5	Middle Crowded	0	1
5,3	04:20 - 04:24	0.5	Middle Crowded	0	1
5,4	04:25 - 04:29	0.5	Middle Crowded	0	1
5,5	04:30 - 04:34	0.5	Middle Crowded	0	1
5,6	04:35 - 04:39	0.5	Middle Crowded	0	1
5,7	04:40 - 04:44	0.5	Middle Crowded	0	1
5,8	04:45 - 04:49	0.5	Middle Crowded	0	1
5,9	04:50 - 04:54	0.5	Middle Crowded	0	1
6	04:55 - 04:59	0.5	Middle Crowded	0	1
6,1	05:00 - 05:04	0.5	Middle Crowded	0	1
6,2	05:05 - 05:09	0.5	Middle Crowded	0	1
6,3	05:10 - 05:14	0.5	Middle Crowded	0	1
6,4	05:15 - 05:19	0.5	Middle Crowded	0	1
6,5	05:20 - 05:24	0.5	Middle Crowded	0	1
6,6	05:25 - 05:29	0.5	Middle Crowded	0	1
6,7	05:30 - 05:34	0.5	Middle Crowded	0	1
6,8	05:35 - 05:39	0.5	Middle Crowded	0	1
6,9	05:40 - 05:44	0.5	Middle Crowded	0	1
7	05:45 - 05:49	0.5	Middle Crowded	0	1
7,1	05:50 - 05:54	0.5	Middle Crowded	0	1
7,2	05:55 - 05:59	0.5	Middle Crowded	0	1
7,3	06:00 - 06:04	0.5	Middle Crowded	0	1
7,4	06:05 - 06:09	0.5	Middle Crowded	0	1
7,5	06:10 - 06:14	0.5	Middle Crowded	0	1
7,6	06:15 - 06:19	0.5	Middle Crowded	0	1
7,7	06:20 - 06:24	0.5	Middle Crowded	0	1
7,8	06:25 - 06:29	0.5	Middle Crowded	0	1
7,9	06:30 - 06:34	0.5	Middle Crowded	0	1
8	06:35 - 06:39	0.5	Middle Crowded	0	1
8,1	06:40 - 06:44	0.5	Middle Crowded	0	1
8,2	06:45 - 06:49	0.5	Middle Crowded	0	1
8,3	06:50 - 06:54	0.5	Middle Crowded	0	1
8,4	06:55 - 06:59	0.5	Middle Crowded	0	1
8,5	07:00 - 07:04	0.5	Middle Crowded	0	1
8,6	07:05 - 07:09	0.5	Middle Crowded	0	1
8,7	07:10 - 07:14	0.5	Middle Crowded	0	1
8,8	07:15 - 07:19	0.5	Middle Crowded	0	1
8,9	07:20 - 07:24	0.5	Middle Crowded	0	1
…..	……..	…..	…….	……	……
0,1	00:00 - 00:04	0.7	Overcrowded	1	0

0,2	00:05 - 00:09	0.7	Overcrowded	1	0
0,3	00:10 - 00:14	0.7	Overcrowded	1	0
0,4	00:15 - 00:19	0.7	Overcrowded	1	0
0,5	00:20 - 00:24	0.7	Overcrowded	1	0
0,6	00:25 - 00:29	0.7	Overcrowded	1	0
0,7	00:30 - 00:34	0.7	Overcrowded	1	0
0,8	00:35 - 00:39	0.7	Overcrowded	1	0
0,9	00:40 - 00:44	0.7	Overcrowded	1	0
1	00:45 - 00:49	0.7	Overcrowded	1	0
1,1	00:50 - 00:54	0.7	Overcrowded	1	0
1,2	00:55 - 00:59	0.7	Overcrowded	1	0
1,3	01:00 - 01:04	0.7	Overcrowded	1	0
1,4	01:05 - 00:09	0.7	Overcrowded	1	0
1,5	01:10 - 01:14	0.7	Overcrowded	1	0
1,6	01:15 - 01:19	0.7	Overcrowded	1	0
1,7	01:20 - 01:24	0.7	Overcrowded	1	0
1,8	01:25 - 01:29	0.7	Overcrowded	1	0
1,9	01:30 - 01:34	0.7	Overcrowded	1	0
2	01:35 - 01:39	0.7	Overcrowded	1	0
2,1	01:40 - 01:44	0.7	Overcrowded	1	0
2,2	01:45 - 01:49	0.7	Overcrowded	1	0
2,3	01:50 - 01:54	0.7	Overcrowded	1	0
2,4	01:55 - 01:59	0.7	Overcrowded	1	0
2,5	02:00 - 02:04	0.7	Overcrowded	1	0
2,6	02:05 - 02:09	0.7	Overcrowded	1	0
2,7	02:10 - 02:14	0.7	Overcrowded	1	0
2,8	02:15 - 02:19	0.7	Overcrowded	1	0
2,9	02:20 - 02:24	0.7	Overcrowded	1	0
3	02:25 - 02:29	0.7	Overcrowded	1	0
3,1	02:30 - 02:34	0.7	Overcrowded	1	0
3,2	02:35 - 02:39	0.7	Overcrowded	1	0
3,3	02:40 - 02:44	0.7	Overcrowded	1	0
3,4	02:45 - 02:49	0.7	Overcrowded	1	0
3,5	02:50 - 02:54	0.7	Overcrowded	1	0
3,6	02:55 - 02:59	0.7	Overcrowded	1	0
3,7	03:00 - 03:04	0.7	Overcrowded	1	0
3,8	03:05 - 03:09	0.7	Overcrowded	1	0
3,9	03:10 - 03:14	0.7	Overcrowded	1	0
4	03:15 - 03:19	0.7	Overcrowded	1	0
4,1	03:20 - 03:24	0.7	Overcrowded	1	0
4,2	03:25 - 03:29	0.7	Overcrowded	1	0
4,3	03:30 - 03:34	0.7	Overcrowded	1	0
4,4	03:35 - 03:39	0.7	Overcrowded	1	0
4,5	03:40 - 03:44	0.7	Overcrowded	1	0
4,6	03:45 - 03:49	0.7	Overcrowded	1	0

4,7	03:50 - 03:54	0.7	Overcrowded	1	0
4,8	03:55 - 03:59	0.7	Overcrowded	1	0
4,9	04:00 - 04:04	0.7	Overcrowded	1	0
5	04:05 - 04:09	0.7	Overcrowded	1	0
5,1	04:10 - 04:14	0.7	Overcrowded	1	0
5,2	04:15 - 04:19	0.7	Overcrowded	1	0
5,3	04:20 - 04:24	0.7	Overcrowded	1	0
5,4	04:25 - 04:29	0.7	Overcrowded	1	0
5,5	04:30 - 04:34	0.7	Overcrowded	1	0
5,6	04:35 - 04:39	0.7	Overcrowded	1	0
5,7	04:40 - 04:44	0.7	Overcrowded	1	0
5,8	04:45 - 04:49	0.7	Overcrowded	1	0
5,9	04:50 - 04:54	0.7	Overcrowded	1	0
6	04:55 - 04:59	0.7	Overcrowded	1	0
6,1	05:00 - 05:04	0.7	Overcrowded	1	0
6,2	05:05 - 05:09	0.7	Overcrowded	1	0
6,3	05:10 - 05:14	0.7	Overcrowded	1	0
6,4	05:15 - 05:19	0.7	Overcrowded	1	0
6,5	05:20 - 05:24	0.7	Overcrowded	1	0
6,6	05:25 - 05:29	0.7	Overcrowded	1	0
6,7	05:30 - 05:34	0.7	Overcrowded	1	0
6,8	05:35 - 05:39	0.7	Overcrowded	1	0
6,9	05:40 - 05:44	0.7	Overcrowded	1	0
7	05:45 - 05:49	0.7	Overcrowded	1	0
7,1	05:50 - 05:54	0.7	Overcrowded	1	0
7,2	05:55 - 05:59	0.7	Overcrowded	1	0
7,3	06:00 - 06:04	0.7	Overcrowded	1	0
7,4	06:05 - 06:09	0.7	Overcrowded	1	0
7,5	06:10 - 06:14	0.7	Overcrowded	1	0
7,6	06:15 - 06:19	0.7	Overcrowded	1	0
7,7	06:20 - 06:24	0.7	Overcrowded	1	0
7,8	06:25 - 06:29	0.7	Overcrowded	1	0
7,9	06:30 - 06:34	0.7	Overcrowded	1	0
8,2	06:45 - 06:49	0.7	Overcrowded	1	0
…..	……..	…..	…….	…..	…..
27,1	22:30 - 22:34	0.7	Overcrowded	1	0
27,2	22:35 - 22:39	0.7	Overcrowded	1	0
27,3	22:40 - 22:44	0.7	Overcrowded	1	0
27,4	22:45 - 22:49	0.7	Overcrowded	1	0
27,5	22:50 - 22:54	0.7	Overcrowded	1	0
27,6	22:55 - 22:59	0.7	Overcrowded	1	0
27,7	23:00 - 23:04	0.7	Overcrowded	1	0
27,8	23:05 - 23:09	0.7	Overcrowded	1	0
27,9	23:10 - 23:14	0.7	Overcrowded	1	0
28	23:15 - 23:19	0.7	Overcrowded	1	0

28,1	23:20 - 23:24	0.7	Overcrowded	1	0
28,2	23:25 - 23:29	0.7	Overcrowded	1	0
28,3	23:30 - 23:34	0.7	Overcrowded	1	0
28,4	23:35 - 23:39	0.7	Overcrowded	1	0
28,5	23:40 - 23:44	0.7	Overcrowded	1	0
28,6	23:45 - 23:49	0.7	Overcrowded	1	0
28,7	23:50 - 23:54	0.7	Overcrowded	1	0
28,8	23:55 - 23:59	0.7	Overcrowded	1	0

Neural network table is called out by below mentioned code. Software selects random rows and trains so until system ensures the completion of learning process regarding minimize of learning error then apllys test process with unselected data. The weights of the neural network is calculated and presented at table table 4.1;

W1	0.007
W2	0.6
W3	0.2

Table 4.1: Neural Netwok Weights

And the graphs for the perfomance is at figure 4.2;

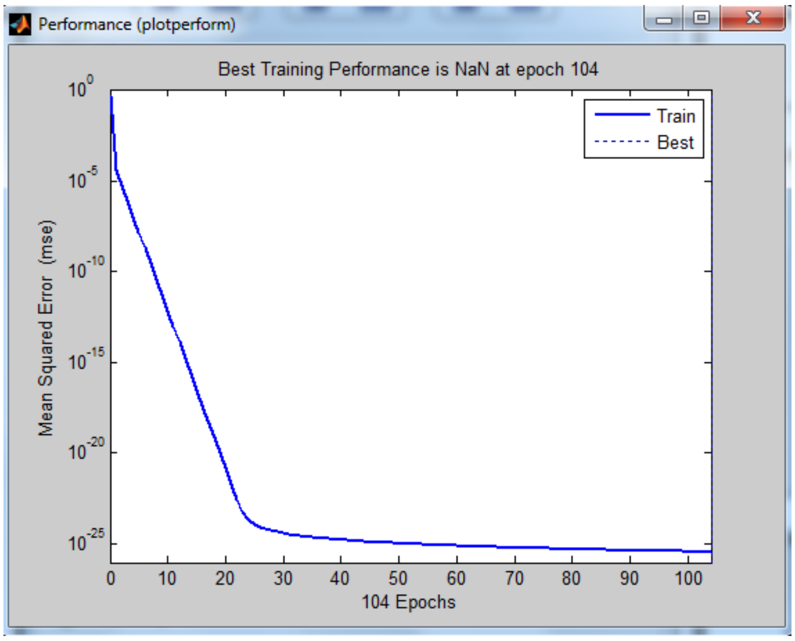

Figure 4.2: Training performance for Neural Network Part

4.2 GENETIC ALGORITHM AS OPTIMIZATION ALGORITHM

After classifying the inputs, system selects one of the GAs to optimize AWT and TTT to the two genetic algoritms;

4.2.1 GENETIC ALGORITHMS PART-1: OPTIMIZING AVERAGE WAITING TIME

With respect to figure 4.1, the algorithm allocates hall calls to perform the elevator group controller in N floors building. The received hall calls are stored into a list according to the moment of the request.

The updated service list of a car consists of the preload condition and the adding stops from the current solution. Define the following parameters to be able to estimate the waiting time of hall call's as following:

- HC; Hall Call Floor
- CF; Car Current Floor
- NF; Number of Building Floors
- TF; Inter-Floor Trip Time

The waiting time of a hall call depends on the direction of hall call up or down, the car trip direction, and also the relative position of hall call related to the current car floor.

As shown in figure 4.3 thru cases are as follows;

Case (I): If Hall Call Floor is up and less than Car Current Floor, the car trip consists of three segments: from Car Current Floor to Top Floor, from Top Floor to 1st floor, and from 1st floor to Hall Call Floor.

Case (II): If Hall Call Floor is down, the car trip consists of two segments: from Car Current Floor to Top Floor and from Top Floor to Hall Call Floor.

Case (III): If Hall Call Floor is up and greater than or equal Car Current Floor, the car trip consists of only one segment: from Car Current Floor to Hall Call Floor.

Therefore, the waiting time Ti of the hall call is given by:

$$T_i = \begin{cases} [HC - CF]\,TF & ; \uparrow HC \geq CF \\ [2\,NF - CF + HC - 2]\,TF & ; \uparrow HC < CF \\ [2\,NF - CF - HC]\,TF & ; \downarrow HC \end{cases} \quad (4.1)$$

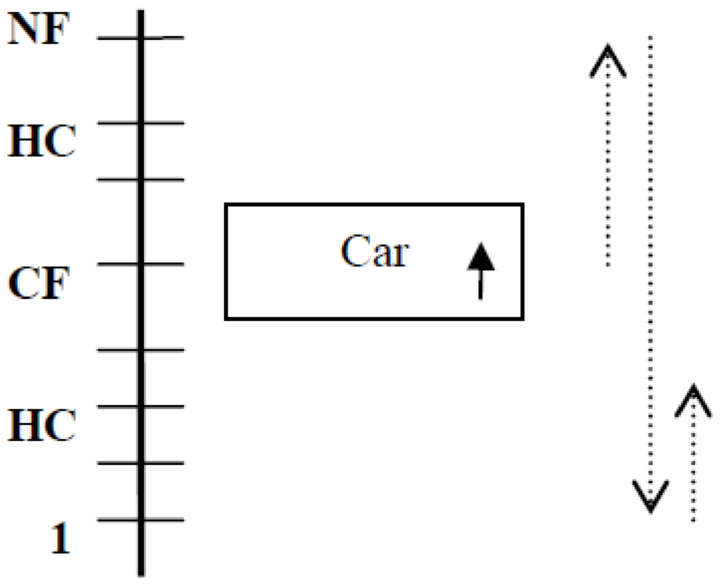

Figure 4.3: Elevator is stopped or going up.

As shown in figure 4.4 thru cases are as follows;

Case (I): If Hall Call Floor is down and greater than Car Current Floor, The car trip of a going down car consists of three segments: from Car Current Floor to 1st floor, from 1st floor to Top Floor, and from Top Floor to Hall Call Floor.

Case (II): If Hall Call Floor is up, the car trip consists of two segments: from Car Current Floor to 1st floor and from 1st floor to Hall Call Floor.

Case (III): If Hall Call Floor is down and less than or equal Car Current Floor, the car trip consists of only one segment: from Car Current Floor to Hall Call Floor.

Therefore, the waiting time Ti of the hall call is given by:

$$T_i = \begin{cases} [CF - HC]\,TF & ; \downarrow HC \le CF \\ [2\,NF + CF - HC - 2]\,TF & ; \downarrow HC > CF \\ [CF + HC - 2]\,TF & ; \uparrow HC \end{cases} \tag{4.2}$$

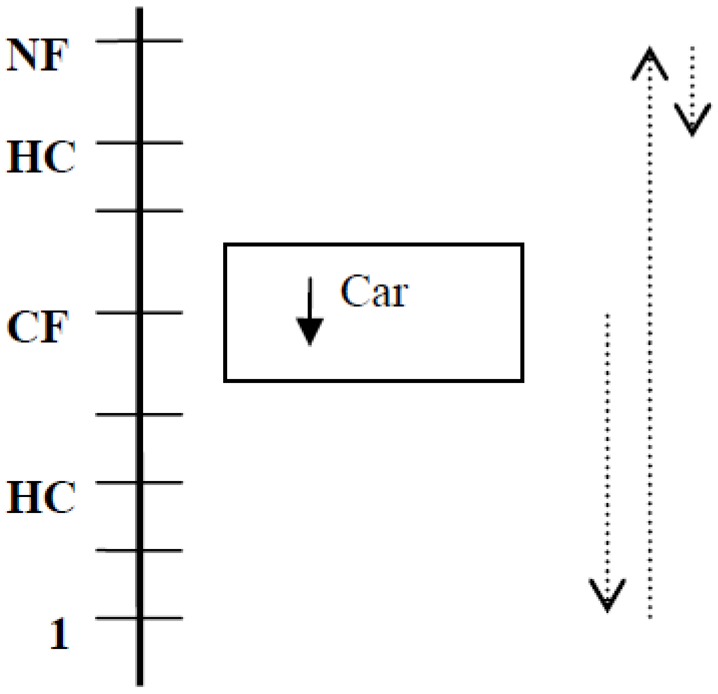

Figure 4.4: Elevator is going down.

Considering optimization algorithm a suitable chromosome must be defined in figure 4.5 presents the main chromosome structure.

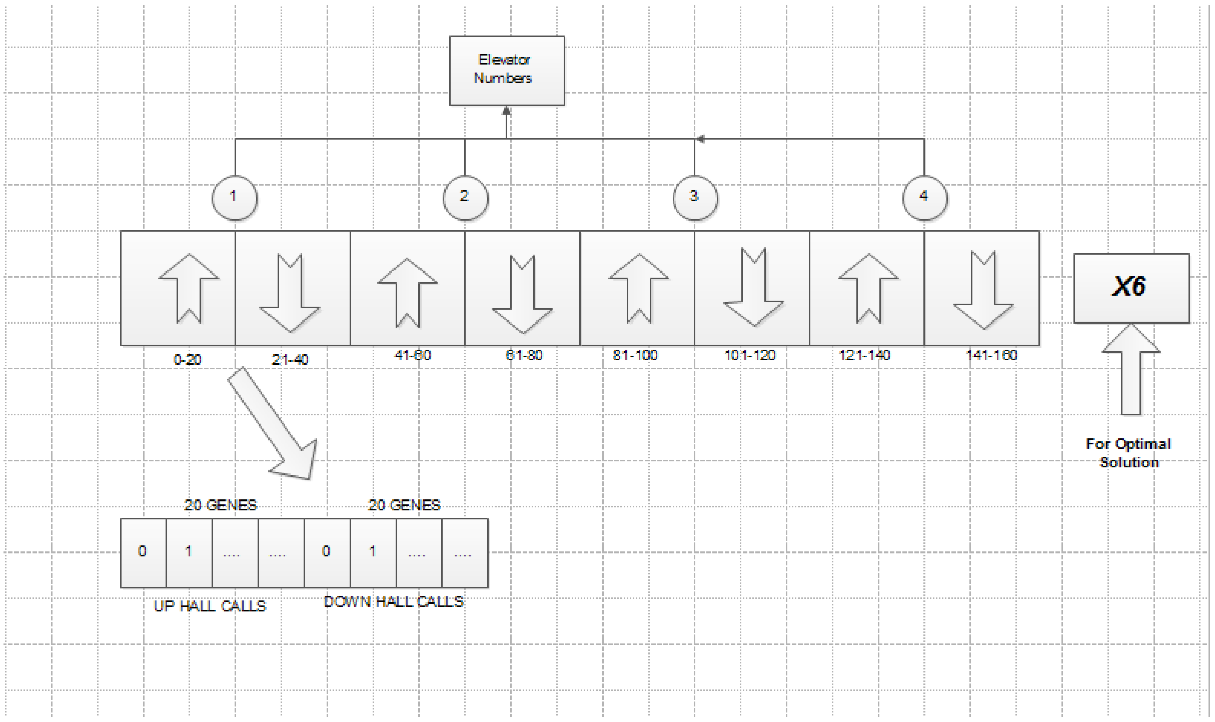

Figure 4.5: Example of a chromosome for an elevator for a 20-floor building

Within the above mentioned chromosome structure, the one chormose has 160 genes for all four elevator gropus. The first 40 genes belonging the first lift is split up from a random 20 genes for up calls and the second 20 genes for down calls following the receipt of hall call information. We choose to generate 6 chromosomes beczues of at least one chromosome converges the intended solution. Randomly splitting chromosome is subjected to crossover process with 0.7 probability and mutation process with 0.2 probability by means of the indigenously designed code. In consequence of these processes, the optimal result of average waiting time is determined. The graphics of the results as shown at figures 4.6, 4.7, 4.8, 4.9, 4.10, 4.11 and 4.12.

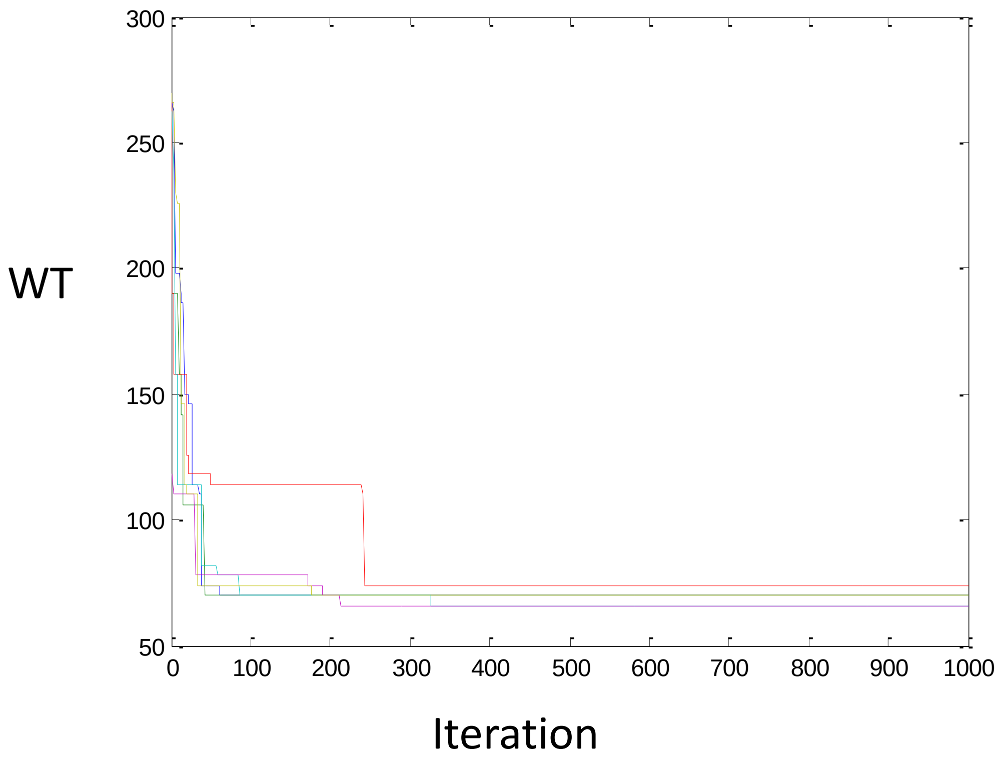

Figure 4.6: The result after 1000 iterations and 6 chromosomes

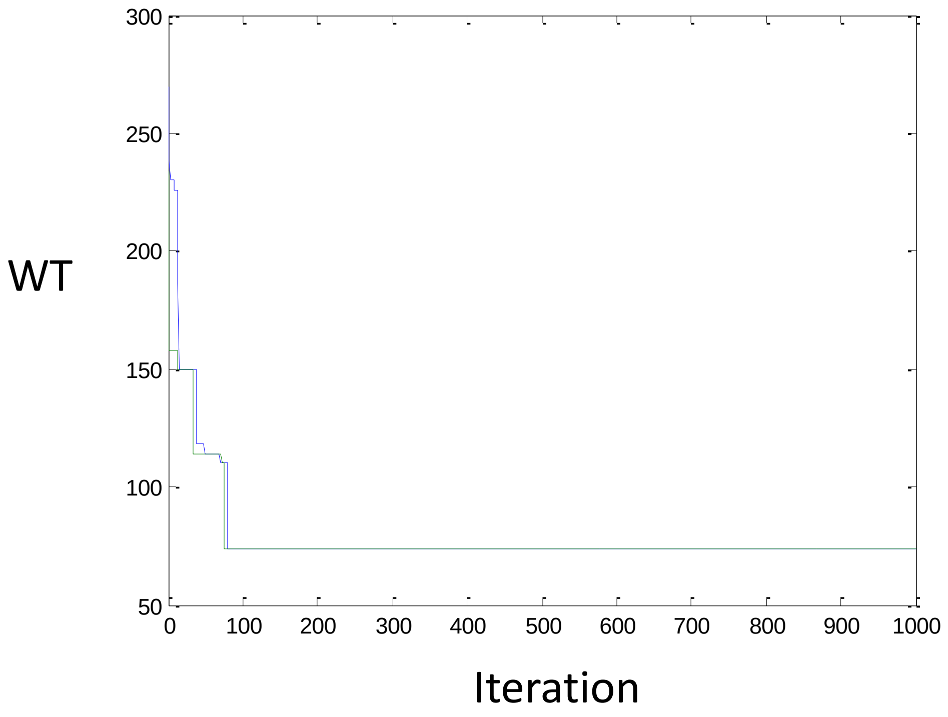

Figure 4.7: The result after 1000 iterations and 2 chromosomes

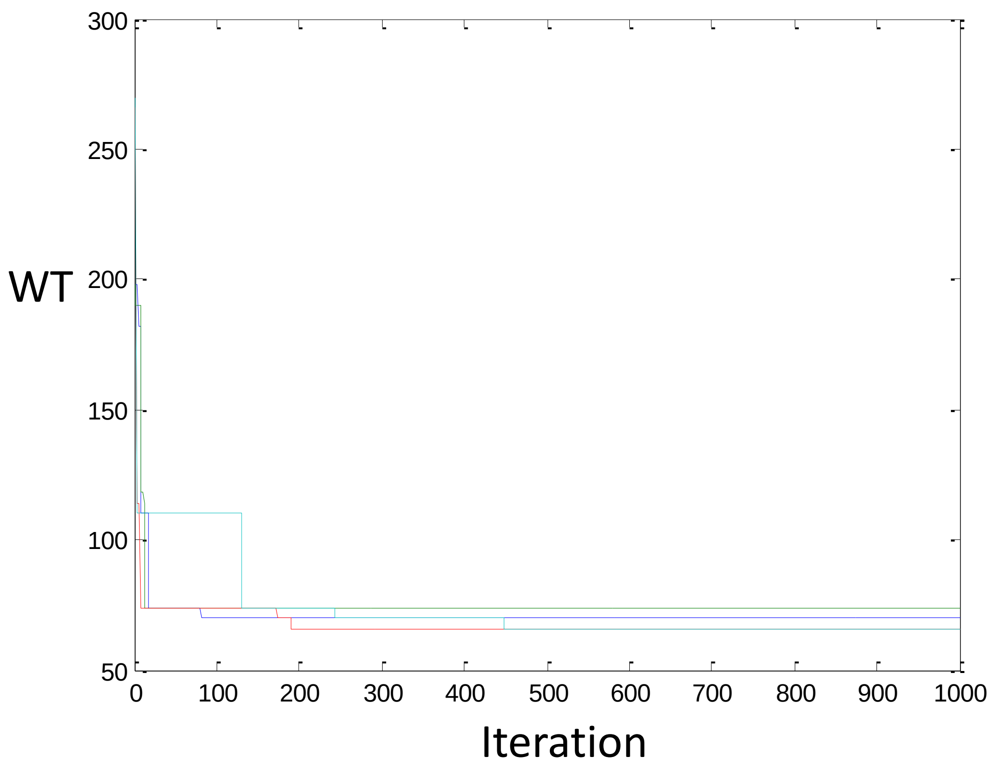

Figure 4.8: The result after 1000 iterations and 4 chromosomes

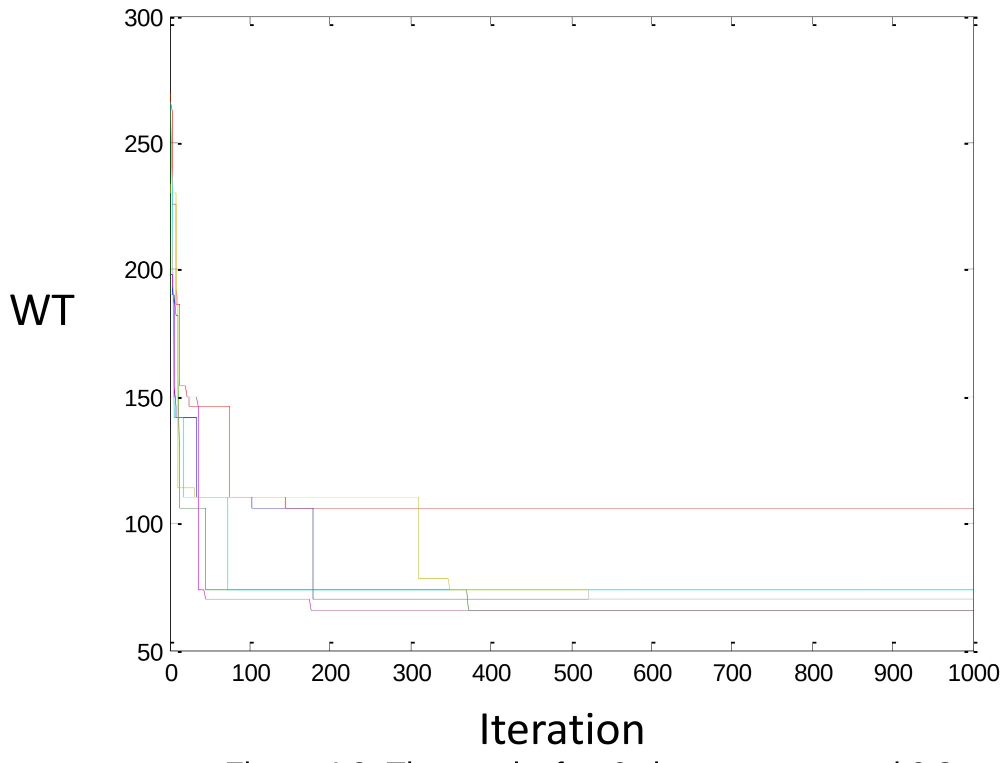

Figure 4.9: The result after 6 chromosomes and 0.8 crossover %

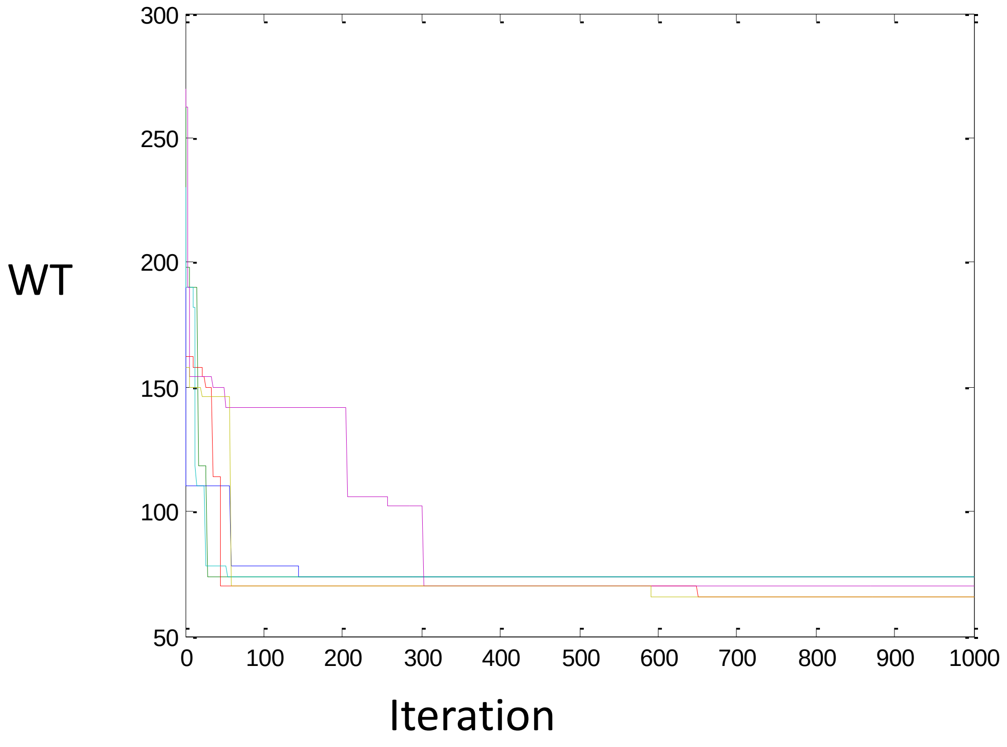

Figure 4.10: The result after 6 chromosomes and 0.9 crossover %

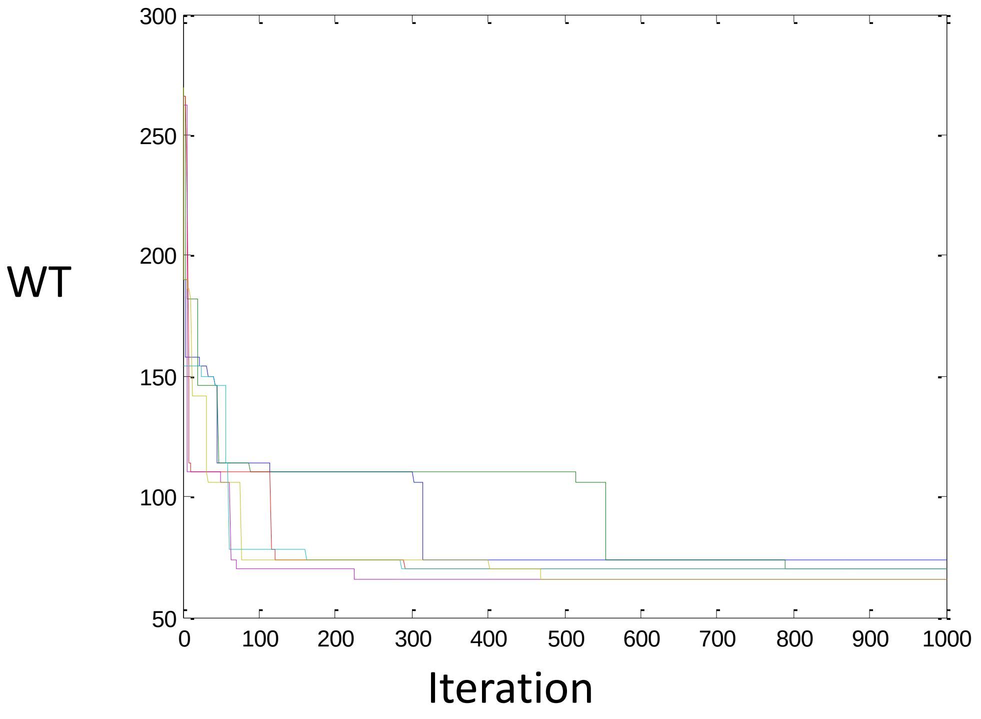

Figure 4.11 The result after 6 chromosomes and 0.1 mutation %

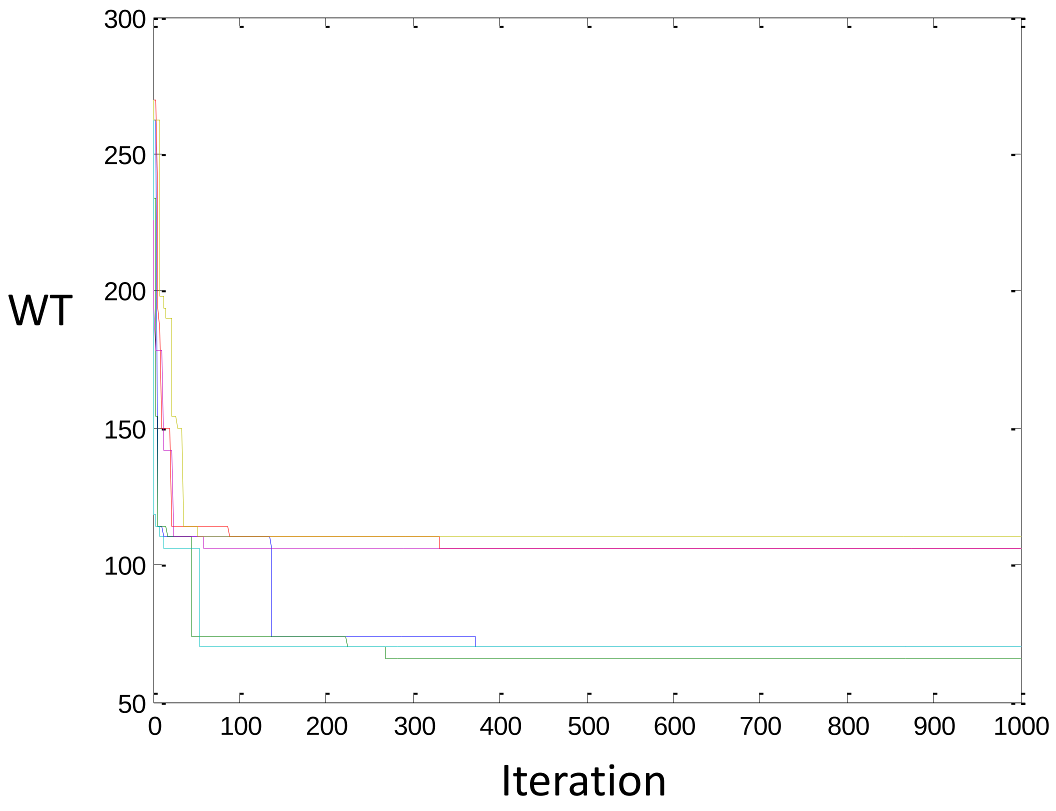

Figure 4.12: The result after 6 chromosomes and 0.01 mutation %

Above graphics show the average waiting time and related calculations with respect to changing chromosome number, crossover and mutation percentages. Increase in the chromosome number and in the probability of crossover and mutation would enable us yielding the positive conclusion in shorter times.

Owing this increase is an optimization problem, optimal chromosome number must be determined. Graphics reveal that the structure of 6-chromosoms with 0.7 crossover and 0.2 mutation probability gives the best result.

4.2.2 GENETIC ALGORITHMS PART-2: OPTIMIZING TOTAL TRAVEL TIME AND ENERGY CONSUMPTION

The fitness function of GA to optimize travel time regarding defined cases which are as follows in figure 4.13;

Case (I): If Hall Call Floor is up and less than Car Current Floor, the car trip consists of three segments: from Car Current Floor to Top Floor, from Top Floor to 1st floor, and from 1st floor to Hall Call Floor.

Case (II): If Hall Call Floor is down, the car trip consists of two segments: from Car Current Floor to Top Floor and from Top Floor to Hall Call Floor.

Case (III): If Hall Call Floor is up and greater than or equal Car Current Floor, the car trip consists of only one segment: from Car Current Floor to Hall Call Floor.

Therefore, the travel time CTj of the hall call is given by:

$$CTj = \left\{ \begin{array}{ll} [CA\mathrm{R\_DF} - CF]TF & If\ no\ HC\ is\ assigned \\ [NF - CF]TF & A\ HC \geq CF\ and\ HC\ is\ up \\ [2NF - CF - 1]TF & A\ \ HC\ is\ down \\ [3NF - CF - 2]TF & A\ HC < CF\ and\ HC\ is\ up \end{array} \right\}$$

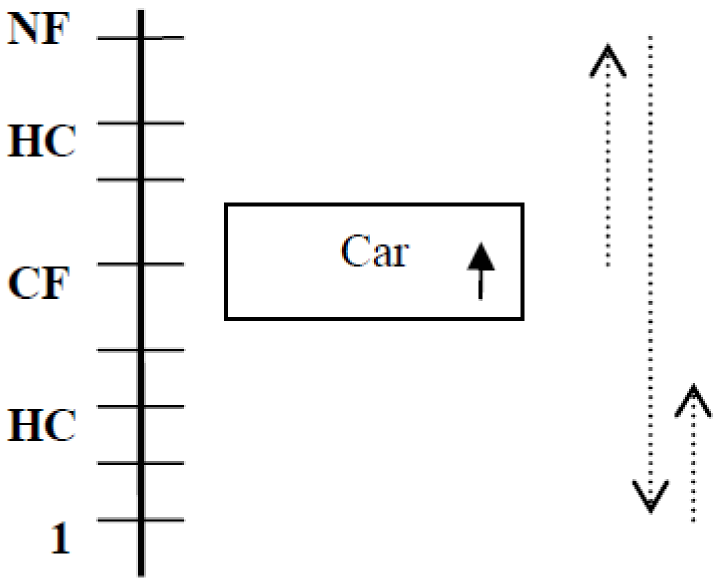

Figure 4.13: Elevator is stopped or going up.

In figure 4.14;

Case (I): If Hall Call Floor is down and greater than Car Current Floor, The car trip of a going down car consists of three segments: from Car Current Floor to 1st floor, from 1st floor to Top Floor, and from Top Floor to Hall Call Floor.

Case (II): If Hall Call Floor is up, the car trip consists of two segments: from Car Current Floor to 1st floor and from 1st floor to Hall Call Floor.

Case (III): If Hall Call Floor is down and less than or equal Car Current Floor, the car trip consists of only one segment: from Car Current Floor to Hall Call Floor.

Therefore, the travel time of CTj of the hall call is given by:

$$CTj = \begin{cases} [CF - CAR\_DF]TF & If\ no\ HC\ is\ assigned \\ [CF - 1]TF & A\ HC \leq CF\ and\ HC\ is\ down \\ [CF + NF - 2]TF & A\ HC\ is\ up \\ [CF - 2NF - 3]TF & A\ HC < CF\ and\ HC\ is\ down \end{cases}$$

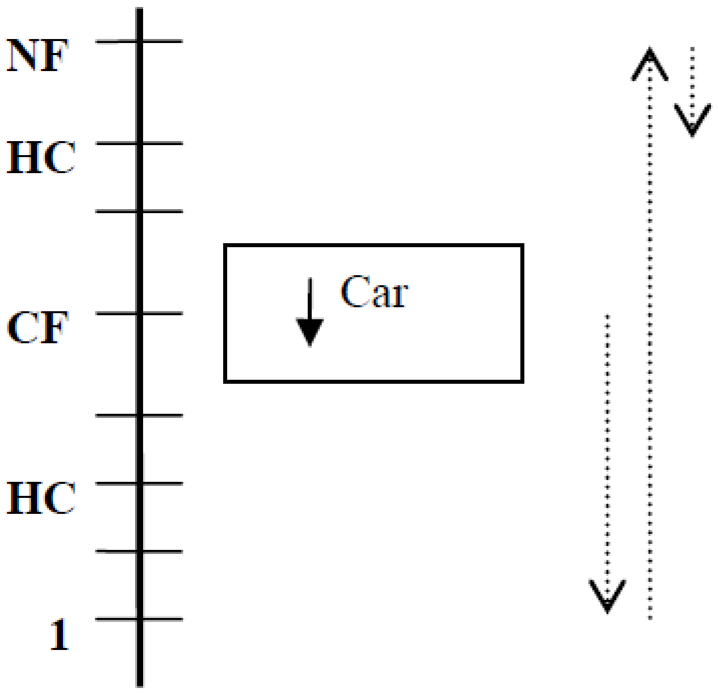

Figure 4.14: Elevator is going down.

.

Randomly splitting chromosome is subjected to crossover process with 0.7 probability and mutation process with 0.2 probability by means of the indigenously designed code. In consequence of these processes, the optimal result of travel time is determined. The graphics of the results as shown at figures 4.15, 4.16, 4.17, 4.18, 4.19, 4.20 and 4.21.

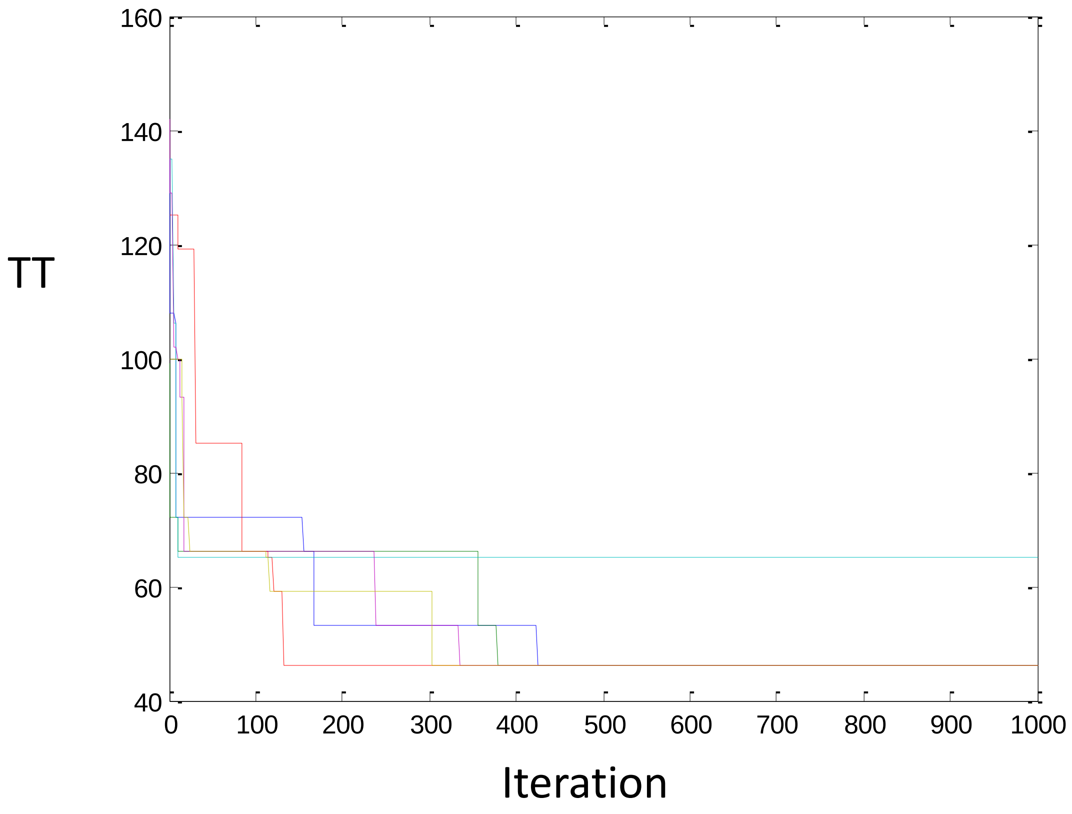

Figure 4.15: The result after 1000 iterations and 6 chromosomes

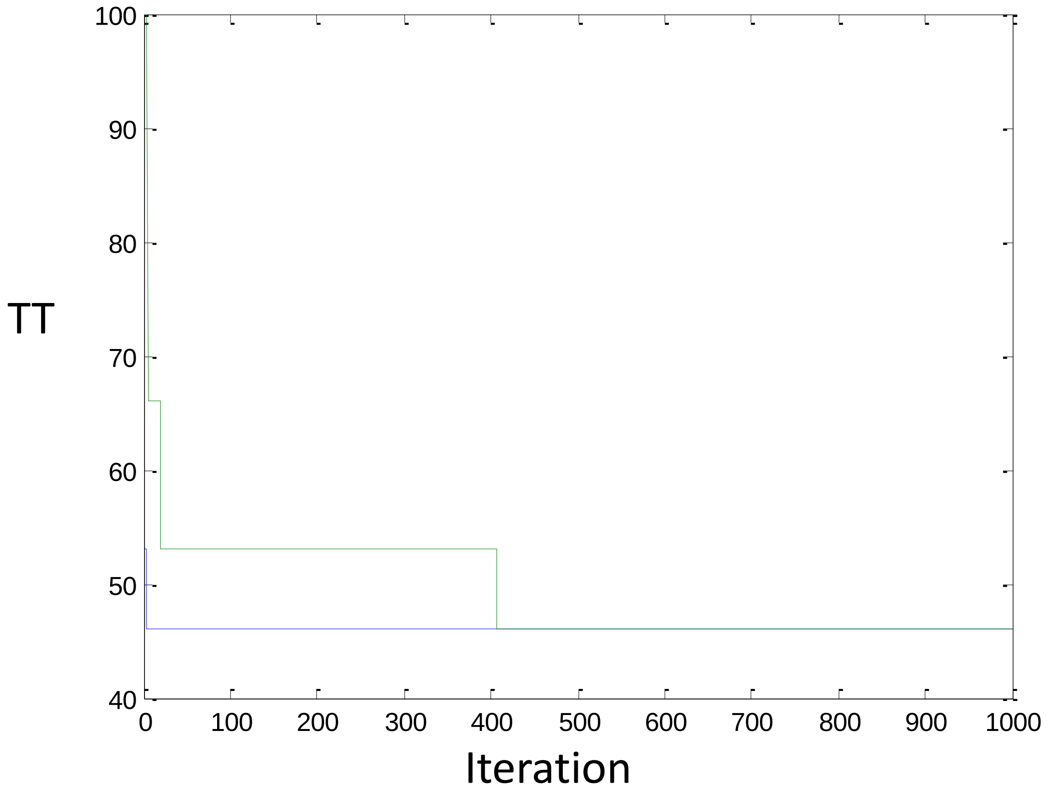

Figure 4.16: The result after 1000 iterations and 2 chromosomes

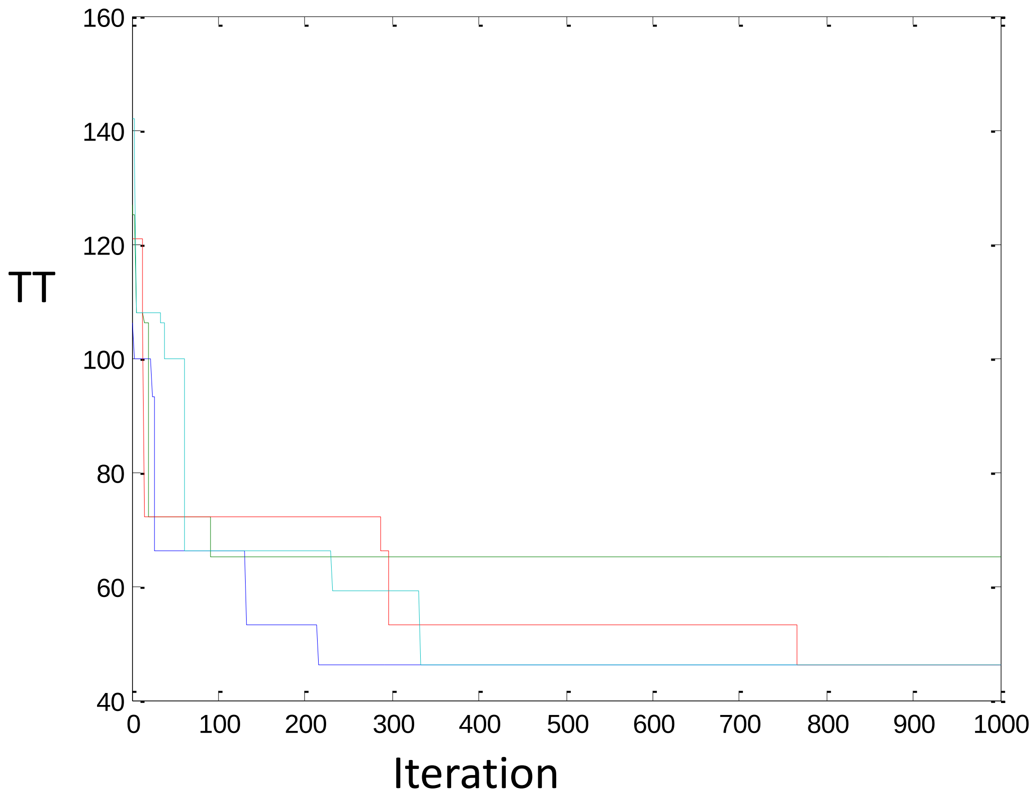

Figure 4.17: The result after 1000 iterations and 4 chromosomes

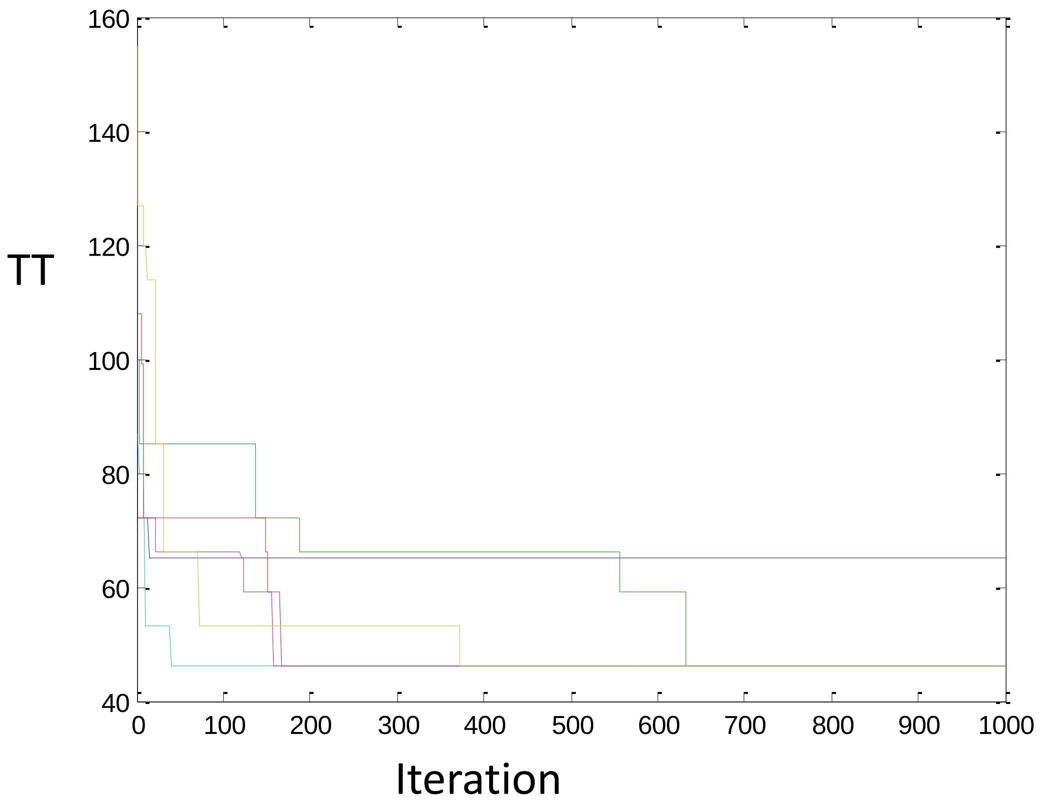

Figure 4.18: The result after 6 chromosomes and 0.7 crossover %

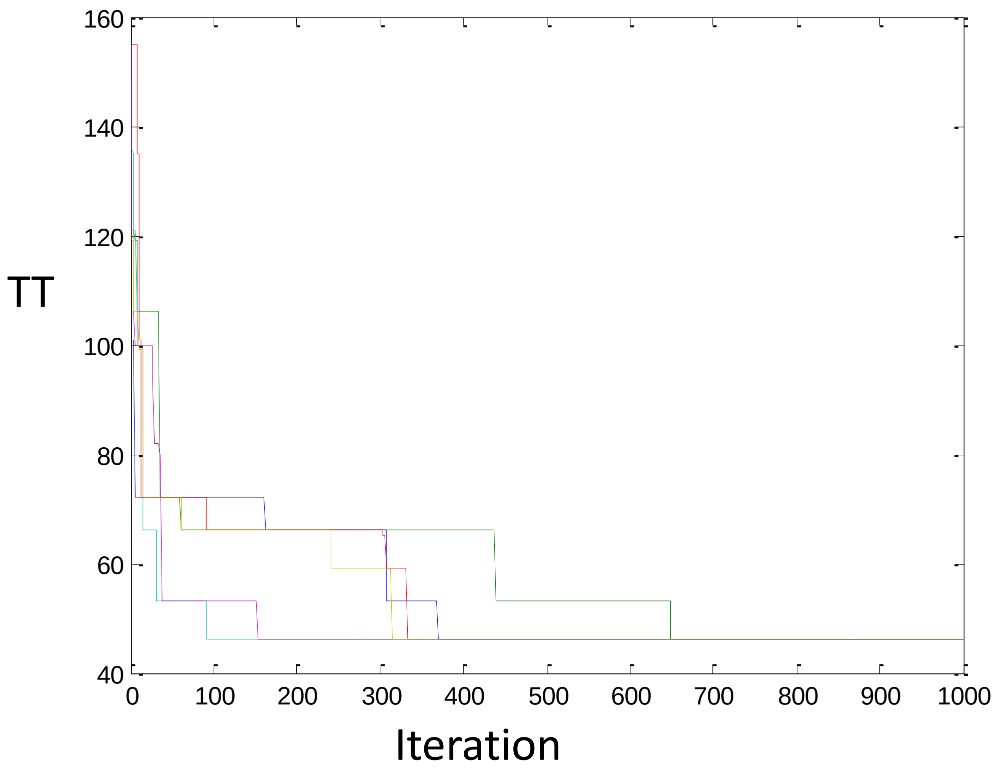

Figure 4.19: The result after 6 chromosomes and 0.9 crossover %

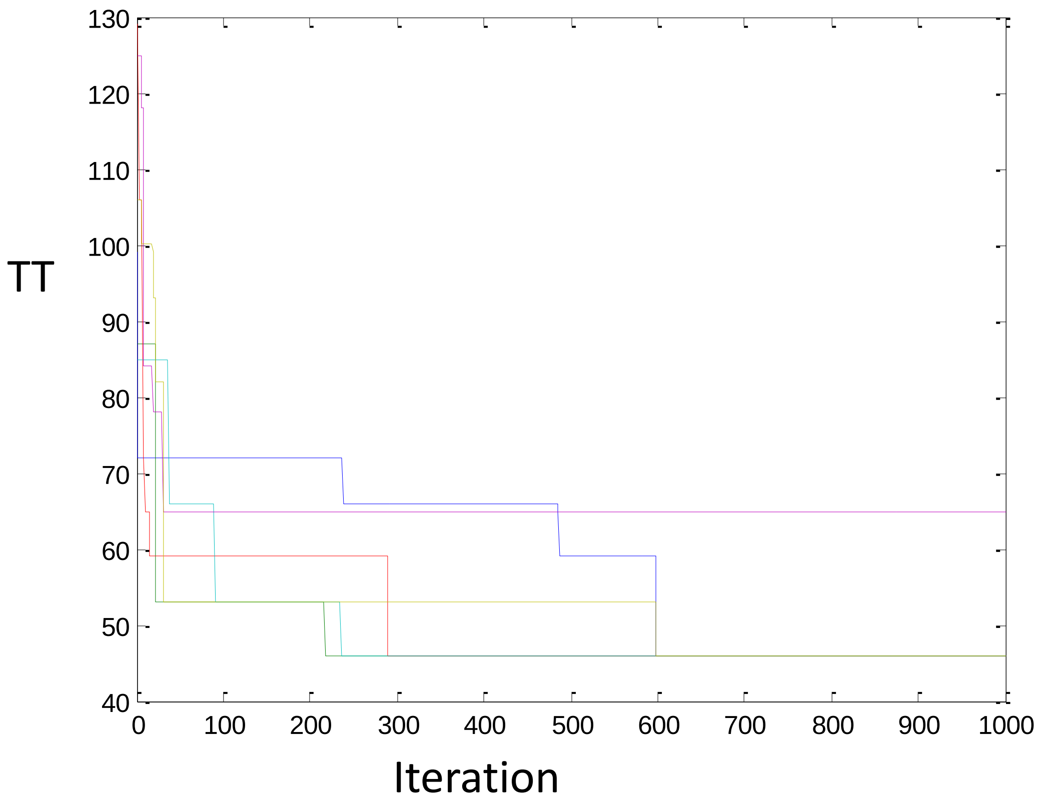

Figure 4.20: The result after 6 chromosomes and 0.2 mutation %

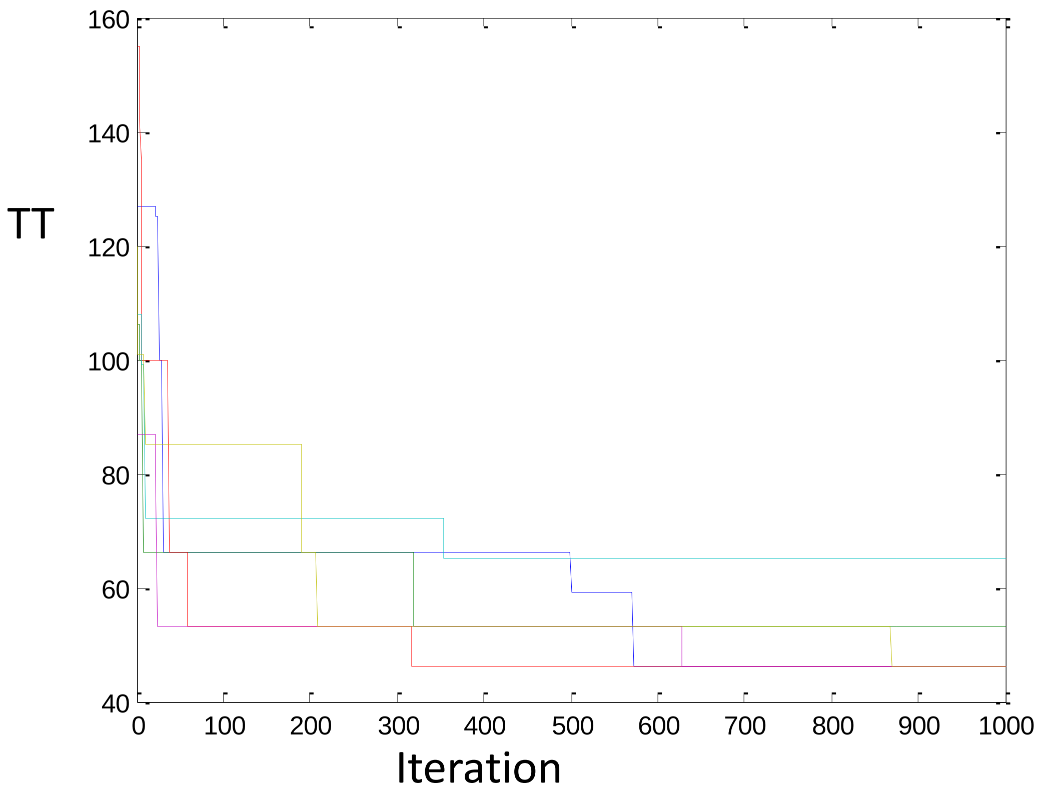

Figure 4.21: The result after 6 chromosomes and 0.01 mutation %

Above graphics show the total travel time & energy consumption and related calculations with respect to changing chromosome number, crossover and mutation percentages. Increase in the chromosome number and in the probability of crossover and mutation would enable us yielding the positive conclusion in shorter times.

Owing this increase is an optimization problem, optimal chromosome number must be determined. Graphics reveal that the structure of 6-chromosoms with 0.7 crossover and 0.2 mutation probability gives the best result.

In genetic algorithm parts, firstly we guided from a study [8] for average waiting time. But this algorithm works all 24 hour in same way and we choose to improve these study for a travel time and energy consumption algorithm. The type of this work is that, after classifying the inputs with artificial neural network our two algorithms works in different conditions. And this type of work makes our project more efficient.

5. SIMULATION

In this project we have two inputs, which are human population and time data. We can select these parameters from our GUI screen which is below at figures at 5.1 and 5.2. By using this GUI, user can select time data, human population and hall calls.

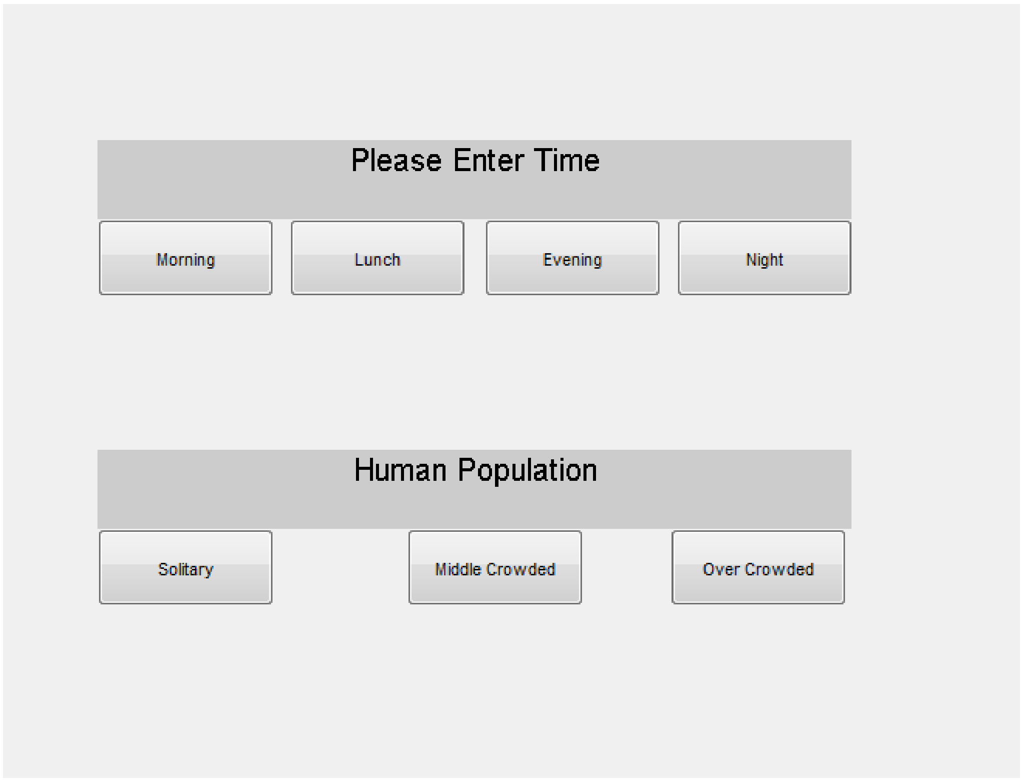

Figure 5.1: Simulation view for Inputs

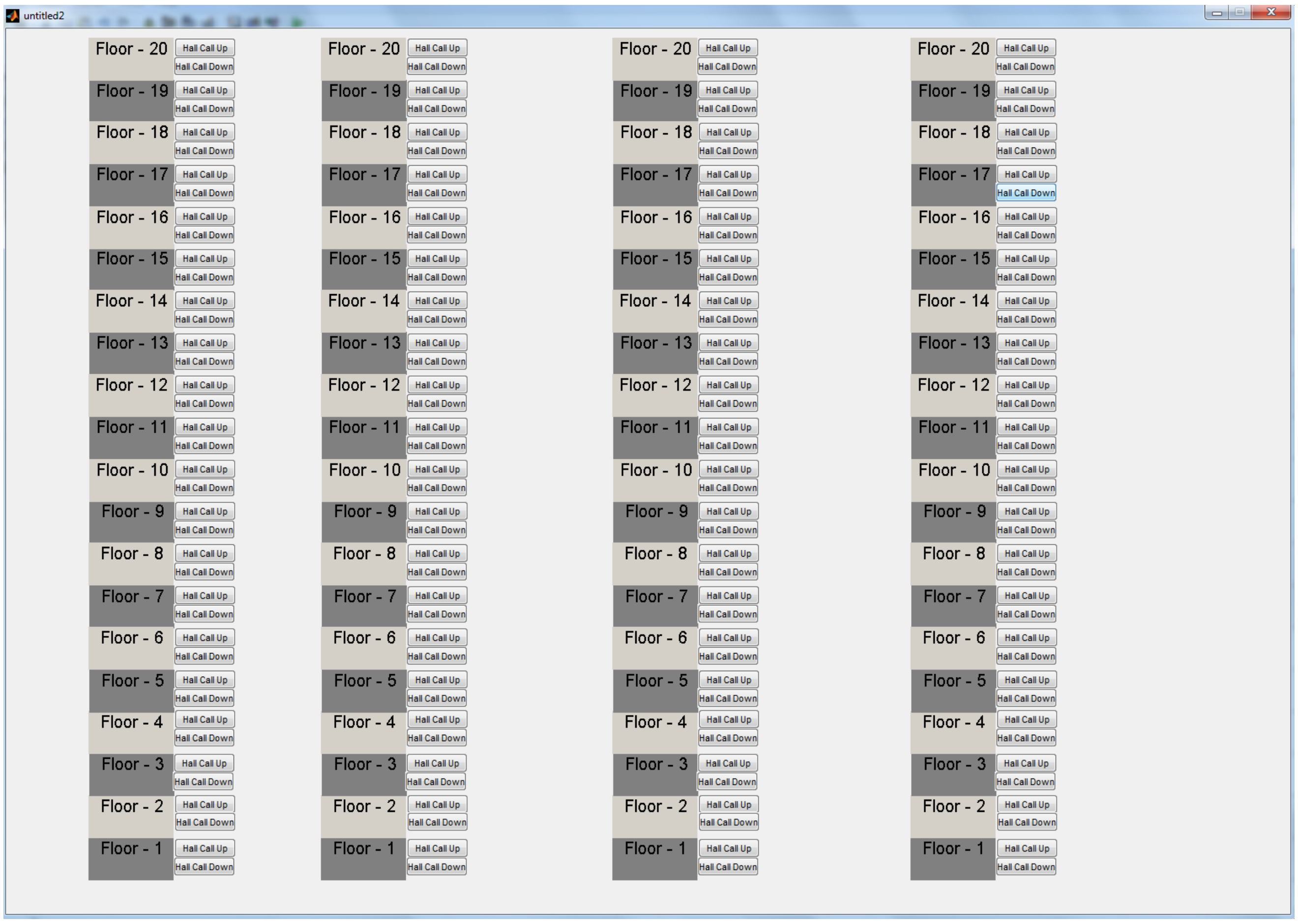

Figure 5.2: Simulation view 20 Floor and 4 elevator structure

Simulation screen is composed of two displays. Input display is used to make day time entries (morning, lunch, evening, night) and the human population (solitary, middle-crowded, over-crowded) which shall draw the program for selection of proper algorithm.

Second screen allows making the hall call entries for a building of 20 floors and 4 cars. Hall call allocations and results are shown by MATLAB screen.

6. EXAMPLES OF THE APPLIED METHOD AND COMPARISE RESULTS

Let us evaluate the observations made after the Average Waiting Time algorithm;

The parameters such as each hall calls, travel directions and population have influence on simulation results.
In order to test the performance of the applied hybrid algorithm a senerio has benn developed for optimazing average waiting time (GA1). The necessary parameters has been selected similiar to [8] which are shown in table 6.2. This senario is as follows;

The values set in that study are applicable for a building of 20 flats with 4 elevators. In this case, floor numbers 7, 11, 13 and 15 claim hall calls to "down" and floor numbers 9 and 12 claim hall calls to "up".

This indigenous Genetic Algorithm method must allocate 1st car to hall calls at floors 9, 12 and 2nd car to hall calls at floors 7, 11, 13, 15 but must not assign any hall call to 3rd and 4th cars. Our study proves to be in compliance with this requirement.

Due to mechanical constraints (table 6.1), default waiting time of the cars in motion is 50sec/car. Below tables 6.2, 6.3 and 6.4 depict the improvement of average waiting time by the help of this study.

Door Opening	2 sec
Door Closing	3 sec
Passenger Transfer	2 sec
Inter-Floor Trip Time	2 sec

Table 6.1: Constants

Algorithm	**Average trip time (sec)**	**Car1 trip time (sec)**	**Car2 trip time (sec)**	**Car3 trip time (sec)**	**Car4 trip time (sec)**
GA [8]	64,5	65	95	48	50
GA	64,5	65	91	48	50

Table 6.2: Cars Trip Time

Algorithm	**Total Stops**	**Car1 Stops**	**Car2 Stops**	**Car3 Stops**	**Car4 Stops**
GA [8]	18	5	9	2	2
GA	18	5	9	2	2

Table 6.3: Car Stops

Hall Calls	**Waiting Time (sec)**
H7	55
H9	22
H11	33
H12	35
H13	22
H15	11
Average Waiting Time	**29.7**

Table 6.4: Waiting Time Analysis

Comparision of the applied algorithm and previous study for travel time is presented in table 6.2. This table gives the same results similiar to table 6.3. The similarity of the obtained reultmatches with the previous study. Additionally the proposed algorithm calculates waiting time regarded the diffirent hall calls. (Table 6.4)

In a second senario for GA1;

In this case, floor numbers 3, 6, 17 and 18 claim hall calls to “up” and floor numbers 7, 5 and 11 claim hall calls to “up”. This indigenous Genetic Algorithm method must allocate 1st car to hall calls at floors 5, 11, 2nd car to hall calls at floors 3and 6, 4th car to hall calls at floors 17, 18 but must not assign any hall call to 3rd car. Our study proves to be in compliance with this requirement as shown at table 6.5.

And the waiting time analysis;

Hall Calls	**Waiting Time (sec)**
H3	49
H5	14
H6	36
H11	33
H17	25
H18	16
Average Waiting Time	**28.83**

Table 6.5: Waiting Time Analysis

Secondly, let us evaluate the observations made after the Total Travel Time and Energy Consumption algorithm;
For case I; Floor numbers 7, 11, 13 and 15 claim hall calls to “down” and floor numbers 9 and 12 claim hall calls to “up”.

Total Floors with Algorithm AWT	Total Floors with Algorithm TTT&EC
67	47

Table 6.6: Floor Trip Analysis

If we use first algorithm all cars goes 67 total floors. But this algoritm (GA2) makes this number to 47 as shown at table 6.6. First and third cars are just maket he job from hall calls, and the second and fourth cars take all other hallcalls. With this algorithm, the total car floors for trip decreases and consequently the consumed energy is reduced.

7. CONCLUSION

This study employs Hybrid Algorithm in the optimization of average waiting time, average travel time and energy consumption in multi-car elevator systems. Hybrid Algorithm is composed of two sections. At the entrance, offlined trained Artificial Neural Network selects one of the optimization algorithm. Second section is the Genetic Algorithm based optimization algorithms that work on-line to optimize parameters. Each optimization algorithms are taken from separate studies. Selection of this algorithm is performed by ANN by taking into account of certain day time intervals and passenger bulge. As the results obtained by single algorithm is compared with the results by hybrid algorithm at the same conditions, it is observed that hybrid algorithm yields better results at different conditions. The main contribute of this study is the selection of the one of the two optimization algorithm with respect to daily conditions and additionaly reducing energy consumption in compare with previous study. The study can be improved by implementing the Fuzzy Logic in lieu of ANN structure, increasing the number of entries and by forming a different simulation structure by using yet developed GUI in C#.

REFERENCES

[1] Bolat Berna, Cortes Pablo, Genetic and tabu search approaches for optimizing the hall call - Car allocation problem in elevator group systems, Applied Soft Computing 11, (2011) 1792–1800.

[2] Alex Valdivielso, Toshiyuki Miyamoto, “Multicar-Elevator Group Control Algorithm for Interference Prevention and Optimal Call Allocation”, IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, Vol. 41, No. 2, March 2011.

[3] Jafferi Jamaludin, Nasrudin Abd. Rahim and Wooi Ping Hew “An Elevator Group Control System With a Self-Tuning Fuzzy Logic Group Controller”, IEEE Transactions on Industrial Electronics, Vol. 57, No. 12, December 2010.

[4] Lu Yu, Shingo Mabu, Kotaro Hirasawa “Multicar Elevator Group Supervisory Control System using Genetic Network Programming”, IEEJ Transactions on Electrical and Electronic Engineering IEEJ Trans 2011; 6(S1): S65–S73.

[5] Jin Zhou, Lu Yu, Shingo Mabu, Kotaro Hirasawa, Kotaro Hirasawa, Sandor Markon, “A Traffic-Flow-Adaptive Controller of Double-Deck Elevator Systems using Genetic Network Programming”, IEEJ Transactions on Electrical and Electronic Engineering IEEJ Trans 2008; 3: 703–714.

[6] J. Jamaludin, N.A.Rahim, W.P.Hew, “Development of a Self-tuning Fuzzy Logic Controller for Intelligent Control of Elevator Systems”, Engineering Applications of Artificial Intelligence 22 (2009) 1167–1178.

[7] Jin Sun, Qian-Chuan Zhao, Peter B. Luh, “Optimization of Group Elevator Scheduling With Advance Information”, IEEE Transactions on Automation Science and Engineering, Vol. 7, No. 2, April 2010.

[8] W. GHARIEB, “Optimal Elevator Group Control Using Genetic Algorithms”, Computer and Systems Engineering Dept., Faculty of Engineering Ain Shams University.

[9] Janne S. Sorsa, Harri Ehtamo, Marja-Liisa Siikonen, Tapio Tyni, Jari Ylinen, “The Elevator Dispatching Problem”, KONE Corporation, Submitted to Transportation Science Manuscript 1, September 2009.

[10] Philipp Friese, Jörg Rambau, “Online-Optimization of Multi-Elevator Transport Systems with Reoptimization Algorithms Based on Set-Partitioning Models”, KONE Corporation, Submitted to Transportation Science Manuscript 1, September 2009 Zuse-Institute Berlin, Takustr. 7, 14195 Berlin, Germany.

[11] Lu Yu, Shingo Mabu, Kotaro Hirasawa, Tsuyoshi Ueno, “Analysis of Energy Consumption of Elevator Group Supervisory Control System Based on Genetic Network Programming”, IEEJ Transactions on Electrical and Electronic Engineering IEEJ Trans 2011; 6: 414–423.

[12] Bo Xiong, Peter B. Luh, Shi Chung Chang, “Group Elevator Scheduling with Advanced Traffic Information for Normal Operations and Coordinated Emergency Evacuation”, Proceedings of the 2005 IEEE International Conference on Robotics and Automation Barcelona, Spain, April 2005.

[13] Thomas Beielstein, Sandor Markon, Mike Preuss, “A Parallel Approach to Elevator Optimization Based on Soft Computing”, MIC2003: The Fifth Metaheuristics International Conference Kyoto, Japan, August 25–28, 2003.

[14] Thomas Bartz-Beielstein, Mike Preuss, Sandor Markon, “Validation and Optimization of an Elevator Simulation Model with Modern Search Heuristics”, Metaheuristics: Progress as Real Problem Solvers.

[15] Marja-Liisa Siikonen, “Planning and Control Models for Elevators in High-Rise Buildings ”, Helsinki University of Technology Systems Analysis Laboratory Research Reports A68 October 1997.

[16] Mirko Ruokokoski, Harri Ehtamo, Janne Sorsa, Marja-Liisa Siikonen , “Elevator Dispatching as Mixed Integer Linear Optimization Problem”, INFORMS Annual Meeting October 2008.

[17] Pablo Cortés, Jesús Muñuzuri, Luis Onieva , "Design and Analysis of a Tool for Planning and Simulating Dynamic Vertical Transport", SIMULATION 2006 82: 255 DOI: 10.1177/0037549706066986 Aug 7, 2006.

[18] Saw Soon King, Omrane Bouketir, "Simulation of a Four-Car Elevator Operation Using MATLAB ", Modern Applied Science Vol. 2, No. 6 November, 2008.

[19] Kumeresan A. Danapalaasıngam, " Design of a Simulator for Elevator Supervisory Group Controller Using Ordinal Structure Fuzzy Reasoning with Context Adaptation" Faculty of Electrical Engineering, Universiti Teknologi Malaysia, November,2005

[20] http://en.wikipedia.org/wiki/Neural_network

[21] RC Chakraborty, "Fundamentals of Neural Networks: AI Course Lecture", June 01,2010.

[22] http://en.wikipedia.org/wiki/Genetic_algorithm